Machine Learning

Machine Learning is the subset of AI which gives the machine the ability to use the stat model to learn from the data

Machine Learning is used to:

- Detect patterns in the dataset
- Find hidden insights from the data
- Automate analytical and statistical model building

Steps of Machine Learning:

1. **Data collection**
2. **Data analysis**
3. **Data wrangling/Feature engineering**
4. **Train/Test Algorithms**
5. **Model Selection**
6. **Hyperparameter Tuning/ Optimisation**
7. **Prediction/Deployment**

Scenario 1: Iris Prediction

You have been provided the 'Iris' dataset. You have to predict species of the flowers based on the given features. You have to predict the 'Class' feature, which contains 3 species, namely, Iris-Setosa, Iris-Virginica, and Iris-Versicolor.

Dataset Description:

The dataset contains 5 features:

sepal length (cm): length of the sepal

sepal width (cm): width of the sepal

petal length (cm): length of the petal

petal width (cm): width of the petal

Class: species of the iris flower

Tasks to be performed:

1. Load the data, check its shape and check for null values - Beginner
2. Handle null values based on the given threshold - Beginner
3. Convert categorical to numerical feature using Label Encoder - Beginner
4. Handle outlier values - Intermediate
5. Plot and analyze Correlation - Beginner
6. Split the dataset for training and testing - Beginner
7. Perform K-Fold cross validation over different classification algorithm - Intermediate
8. Train a logistic regression model and perform prediction on test data - Beginner
9. Evaluate the model using confusion matrix and F1-score - Beginner

Topics Covered:

Data collection

Data analysis

Data wrangling/Feature engineering

Train/Test Algorithms

Predicting using the trained model

Evaluating a model: F1-score and Confusion matrix

In [1]:
```
!wget https://www.dropbox.com/s/webw4cr5dsnm3jv/iris1.csv
```

```
--2020-07-22 07:58:29--  https://www.dropbox.com/s/webw4cr5dsnm3jv/iris1.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.3.1, 2620:100:6018:1::a27d:301
Connecting to www.dropbox.com (www.dropbox.com)|162.125.3.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/webw4cr5dsnm3jv/iris1.csv [following]
--2020-07-22 07:58:29--  https://www.dropbox.com/s/raw/webw4cr5dsnm3jv/iris1.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucd3193357c5133902ff317ef8a5.dl.dropboxusercontent.com/cd/0/inline/A7_XRZFM6Ww5Yilj60mOfk38obyVxmZz
k7POB7g7kM_KP8OK--zzxitCb_fuEa0m20J3GQTvSdN55ms9UypBHmSF96UuNz12HGzLh2R0uMgwcl-i_Fk0BZVcKUq5JOrKeyg/file# [following]
--2020-07-22 07:58:30--  https://ucd3193357c5133902ff317ef8a5.dl.dropboxusercontent.com/cd/0/inline/A7_XRZFM6Ww5Yilj6
0mOfk38obyVxmZzk7POB7g7kM_KP8OK--zzxitCb_fuEa0m20J3GQTvSdN55ms9UypBHmSF96UuNz12HGzLh2R0uMgwcl-i_Fk0BZVcKUq5JOrKeyg/fi
le
Resolving ucd3193357c5133902ff317ef8a5.dl.dropboxusercontent.com (ucd3193357c5133902ff317ef8a5.dl.dropboxusercontent.
com)... 162.125.3.15, 2620:100:6018:15::a27d:30f
Connecting to ucd3193357c5133902ff317ef8a5.dl.dropboxusercontent.com (ucd3193357c5133902ff317ef8a5.dl.dropboxuserconte
nt.com)|162.125.3.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4332 (4.2K) [text/plain]
Saving to: ‘iris1.csv’

iris1.csv           100%[===================>]   4.23K  --.-KB/s    in 0s

2020-07-22 07:58:30 (521 MB/s) - ‘iris1.csv’ saved [4332/4332]
```

Question-1: Load and analyze the data

Tasks to do:

Load the data in a pandas DataFrame

Have a look at the first five rows

Check if the dataset contains any null values

Check the shape of the dataset

In [2]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('/content/iris1.csv')
df.head()
```

Out[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Class
0	5.1	3.5	1.4	0.2	Iris-Setosa
1	4.9	3.0	1.4	0.2	Iris-Setosa
2	4.7	3.2	1.3	0.2	Iris-Setosa
3	4.6	3.1	1.5	0.2	Iris-Setosa
4	5.0	NaN	1.4	0.2	Iris-Setosa

In [3]:
```
print('the shape of the data is', df.shape)
```

```
the shape of the data is (150, 5)
```

In [4]:
```
print('The null values in each are:', df.isnull().sum())
```

```
The null values in each are: sepal length (cm)     0
sepal width (cm)     30
petal length (cm)    68
petal width (cm)      0
Class                 0
dtype: int64
```

Question-2: Handle null values

Tasks to do:

Handle null values

if a column has more than 40% null values, drop that column

else fill the null values with mean of that column

In [5]:
```
for column in list(df.columns):
  if df[column].isnull().sum()>(0.40*150):
    df.drop(columns=column, axis=1,inplace=True)
  elif df[column].isnull().sum():
    df[column].replace(np.nan,df[column].mean(),inplace=True)
  else:
    continue
```

In [6]:
```
df
```

Out[6]:

	sepal length (cm)	sepal width (cm)	petal width (cm)	Class
0	5.1	3.5000	0.2	Iris-Setosa
1	4.9	3.0000	0.2	Iris-Setosa
2	4.7	3.2000	0.2	Iris-Setosa
3	4.6	3.1000	0.2	Iris-Setosa
4	5.0	3.0575	0.2	Iris-Setosa
...	...	...	...	...
145	6.7	3.0000	2.3	Iris-Virginica
146	6.3	2.5000	1.9	Iris-Virginica
147	6.5	3.0000	2.0	Iris-Virginica
148	6.2	3.0575	2.3	Iris-Virginica
149	5.9	3.0000	1.8	Iris-Virginica

150 rows × 4 columns

Question-3: We cannot use string objects for prediction, so convert categorical feature to numerical feature

Tasks to do:

Convert the categorical features to numerical values using Label Encoder from sklearn

In [7]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Class']=le.fit_transform(df['Class'])
print(df.head())
print(df['Class'].unique())
```

```
   sepal length (cm)  sepal width (cm)  petal width (cm)  Class
0                5.1            3.5000               0.2      0
1                4.9            3.0000               0.2      0
2                4.7            3.2000               0.2      0
3                4.6            3.1000               0.2      0
4                5.0            3.0575               0.2      0
[0 1 2]
```

Question-4: Handle outlier

Tasks to do:

Check for outlier in all the columns using boxplot.

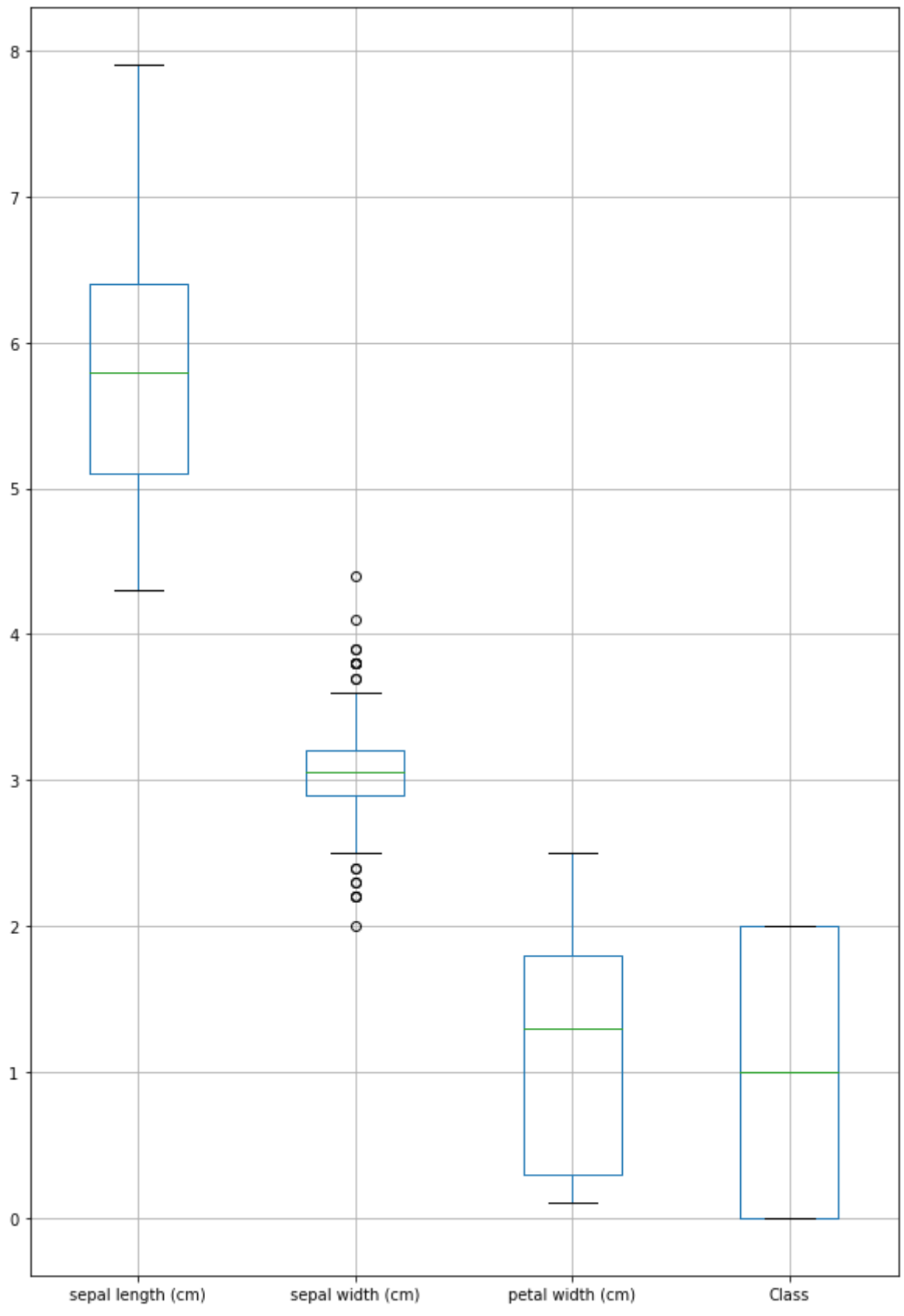
If there are outliers, clip them, lower limit will be Q1-1.5 *IQR and upper limit will be Q3+1.5* IQR.

Q1 = 1st Quartile (25%)

Q3 = 3rd Quartile (75%)

IQR = Inter-quartile range (Q3-Q1)

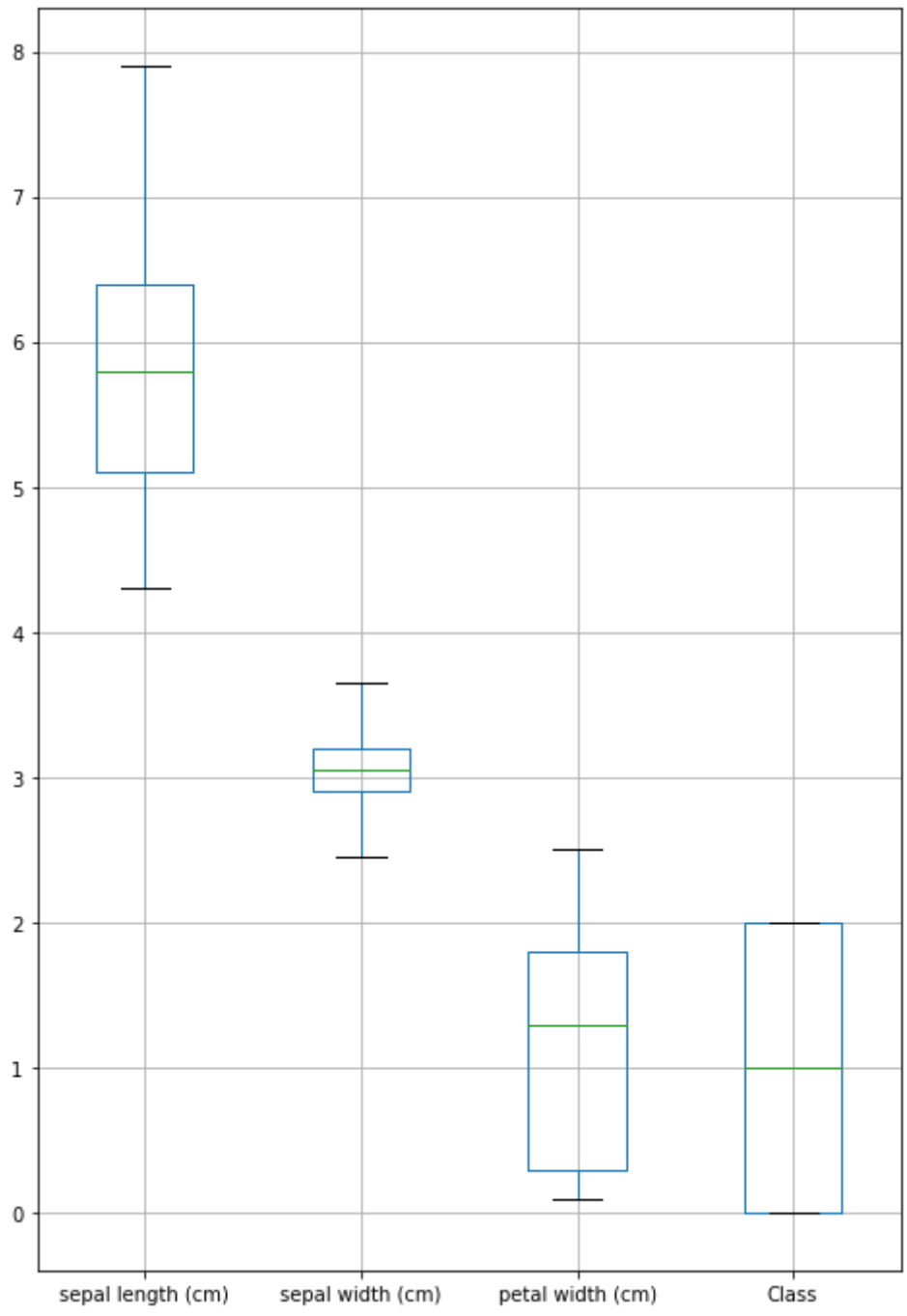
In [8]:
```
import matplotlib.pyplot as plt

df.boxplot(figsize=(10,15))

plt.show()
```



sepal width contains outliers, so we will need to cap them

In [9]:
```
q1 = df['sepal width (cm)'].quantile(.25)
q3 = df['sepal width (cm)'].quantile(.75)
IQR = q3 - q1
df['sepal width (cm)'] = np.clip(df['sepal width (cm)'], q1 - 1.5 * IQR, q3 + 1.5 * IQR)
```

In [10]:
```
df.boxplot(figsize=(8,12))
plt.show()
```



Now we can see there are no outliers left

Question-5: Plot the correlation and tell which feature will help the most while prediction

Calculate correlation

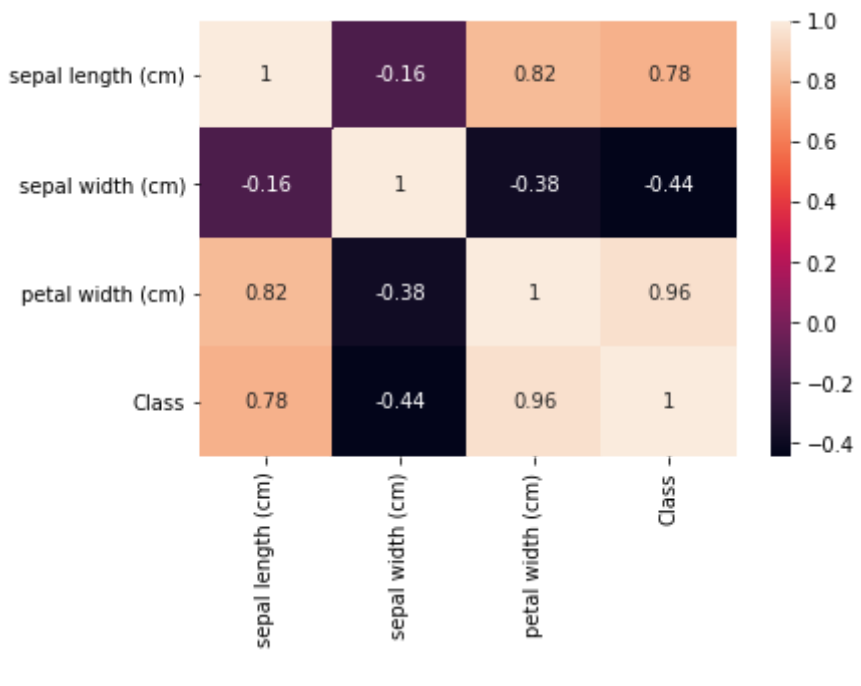
Plot the correlation

Compare the correlation

In [11]:
```
import seaborn as sns
correlation = df.corr()
sns.heatmap(correlation, annot = True)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
ated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb52a86f4a8>



Petal and sepal width are highly correlated with feature Class

Question-6: Split the data into training and testing datasets

Tasks to do:

> Split the dataset using sklearn, with 20% for testing with random_state=7

In [12]:
```
from sklearn.model_selection import train_test_split
X= df.iloc[:,:-1]
y= df.iloc[:,-1]
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.20,random_state = 7)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(120, 3)
(30, 3)
(120,)
(30,)
```

Question-7: Perform K-Fold cross validation

Tasks to do:

> Perform K-fold with K=10 with random_state = 7
>
> Perform K-Fold with commonly used classification algorithm
>
> Calculate the mean score of each iteration

In [13]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [14]:
```
from sklearn.model_selection import cross_val_score,KFold
#machine learning algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

models=[]
models.append(('lr',LogisticRegression()))
models.append(('decision tree',DecisionTreeClassifier()))
models.append(('svm',SVC(gamma='auto')))
models.append(('knn',KNeighborsClassifier()))
models.append(('naive bayes',GaussianNB()))
models.append(('Random Forest',RandomForestClassifier()))


for name,model in models:
  kfold=KFold(n_splits=10,random_state=7)
  cross_val_sc=cross_val_score(model,X,y,scoring='accuracy',cv=kfold)
  print('{} : acc: {}(standard deviation: {})'.format(name,cross_val_sc.mean(),cross_val_sc.std()))
```

```
lr : acc: 0.9200000000000002(standard deviation: 0.07774602526460399)
decision tree : acc: 0.9200000000000002(standard deviation: 0.07774602526460399)
svm : acc: 0.9333333333333333(standard deviation: 0.07302967433402213)
knn : acc: 0.9333333333333333(standard deviation: 0.06666666666666665)
naive bayes : acc: 0.9400000000000001(standard deviation: 0.06289320754704401)
Random Forest : acc: 0.9333333333333333(standard deviation: 0.08432740427115676)
```

Question-8: Train the model

Tasks to do:

Train a logistic regression model for prediction

Also, predict the classes for test data

In [15]:
```
model= LogisticRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
```

Question-9: Evaluate the model

Print confusion matrix of the test data

Also, find the precision and recall using classification report

- precision is the fraction of relevant instances among the retrieved instances
- while recall (also known as sensitivity) is the fraction of the total number of relevant instances retrieved
- https://en.wikipedia.org/wiki/Precision_and_recall (https://en.wikipedia.org/wiki/Precision_and_recall) visit this link to understand further

In [16]:
```
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[7 0 0]
 [0 9 3]
 [0 2 9]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.82      0.75      0.78        12
           2       0.75      0.82      0.78        11

    accuracy                           0.83        30
   macro avg       0.86      0.86      0.86        30
weighted avg       0.84      0.83      0.83        30
```

Scenario 2: Beer Consumption Prediction

The data (sample) were collected in São Paulo—Brazil, in a university, where there are some parties with groups of students from 18 to 28 years of age (average). The dataset used for this activity has 7 attributes, being a Target, with a period of one year. You have to predict the quantity of beer consumption based on the features that contain climate conditions.

Dataset Description:

The dataset contains 7 features:

Data: date of the record

Temperatura Media (C): Average temperature of the day in celsius

Temperatura Minima (C): Minimum temperature of the day in celsius

Temperatura Maxima (C): Maximum temperature of the day in celsius

Precipitacao (mm): Percipitation in mm

Final de Semana: If the day is weekend or not

Consumo de cerveja (litros): Beer consumption in liters

Tasks to be performed:

1. Load the dataset, check its shape, Perform EDA using Pandas Profiling - Intermediate
2. Rectify the data of the first four columns - Intermediate
3. Create new features using the 'Data' feature and the make 'Data' column as index - Intermediate
4. Handle null and duplicate values - Beginner
5. Check the data-type of the features and convert them to appropriate data type - Beginner
6. Analyze features with outlier values - Intermediate
7. Plot and analyze correlation - Beginner
8. Split the dataset for training and testing - Beginner
9. Train a linear regression model and print the intercept and coefficients - Beginner
10. Evaluate the model using R2 score, mean absolute error, and root mean squared error - Beginner

Topics Covered:

Data collection

Data analysis

Data wrangling/Feature engineering

Train/Test Algorithms

Predicting using the trained model

Evaluating a model: R2-score, Mean Absolute Error, and root mean squared error

fetch and download the data

In [17]:
```
!wget https://www.dropbox.com/s/9tmnvhivvq4oyc7/Consumo_cerveja.csv
```

```
--2020-07-22 07:58:36--  https://www.dropbox.com/s/9tmnvhivvq4oyc7/Consumo_cerveja.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.3.1, 2620:100:6018:1::a27d:301
Connecting to www.dropbox.com (www.dropbox.com)|162.125.3.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/9tmnvhivvq4oyc7/Consumo_cerveja.csv [following]
--2020-07-22 07:58:37--  https://www.dropbox.com/s/raw/9tmnvhivvq4oyc7/Consumo_cerveja.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uca4ec71a8495a6b1f56dd7b9de1.dl.dropboxusercontent.com/cd/0/inline/A7-0TcU7vakhxedRk7HZLsaa5XjguN-g
YogjZKWO3LoFfUc04NcIcU56bi6XD3oj5UikiWWtZDYkHCR-HKvORBB4RuXw_Bu4eLDQfEglvCgeF_2S-9XvP2rEnWvizRWZVIo/file# [following]
--2020-07-22 07:58:37--  https://uca4ec71a8495a6b1f56dd7b9de1.dl.dropboxusercontent.com/cd/0/inline/A7-0TcU7vakhxedRk
7HZLsaa5XjguN-gYogjZKWO3LoFfUc04NcIcU56bi6XD3oj5UikiWWtZDYkHCR-HKvORBB4RuXw_Bu4eLDQfEglvCgeF_2S-9XvP2rEnWvizRWZVIo/fi
le
Resolving uca4ec71a8495a6b1f56dd7b9de1.dl.dropboxusercontent.com (uca4ec71a8495a6b1f56dd7b9de1.dl.dropboxusercontent.
com)... 162.125.3.15, 2620:100:6018:15::a27d:30f
Connecting to uca4ec71a8495a6b1f56dd7b9de1.dl.dropboxusercontent.com (uca4ec71a8495a6b1f56dd7b9de1.dl.dropboxusercont
ent.com)|162.125.3.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20376 (20K) [text/plain]
Saving to: ‘Consumo_cerveja.csv’

Consumo_cerveja.csv 100%[===================>]  19.90K  --.-KB/s    in 0.01s

2020-07-22 07:58:37 (2.01 MB/s) - ‘Consumo_cerveja.csv’ saved [20376/20376]
```

Question-1: Load and anlayze the dataset

Tasks to do:

> Load the data in a pandas DataFrame
>
> Have a look at the first five rows

In [18]:
```
import pandas as pd
df1 = pd.read_csv('/content/Consumo_cerveja.csv',parse_dates=['Data'])
```

Question-2: Rectify the data

Tasks to do:

> Replace ',' with '.' in columns 'Temperatura Media (C)', 'Temperatura Minima (C)', 'Temperatura Maxima (C)', and 'Precipitacao (mm)'.
>
> Then check if the data contains appropriate values now

In [19]:
```
df1['Temperatura Media (C)']=df1['Temperatura Media (C)'].str.replace(',','.')
df1['Temperatura Minima (C)'] = df1['Temperatura Minima (C)'].str.replace(',','.')
df1['Temperatura Maxima (C)'] = df1['Temperatura Maxima (C)'].str.replace(',','.')
df1['Precipitacao (mm)'] = df1['Precipitacao (mm)'].str.replace(',','.')
```

In [20]:
```
df1.head()
```

Out[20]:

	Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
0	2015-01-01	27.3	23.9	32.5	0	0.0	25.461
1	2015-01-02	27.02	24.5	33.5	0	0.0	28.972
2	2015-01-03	24.82	22.4	29.9	0	1.0	30.814
3	2015-01-04	23.98	21.5	28.6	1.2	1.0	29.799
4	2015-01-05	23.82	21	28.3	0	0.0	28.900

Question-3: Create new features using the 'Data' feature and make the 'Data' column as index

Tasks to do:

Create new feature 'Month' from the dates, consisting of the month of the year

Create new feature 'Day' from the dates, consisting of the day of the week

Set values from 'Data' column as indexes

In [21]:
```
df1['Month']=df1.Data.dt.month
df1['day']=df1.Data.dt.dayofweek
df1.iloc[335:341]
```

Out[21]:

	Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)	Month	day
335	2015-12-02	22.1	18.2	29.4	0	0.0	30.471	12.0	2.0
336	2015-12-03	22.44	20.2	26.1	0	0.0	28.405	12.0	3.0
337	2015-12-04	22.76	19	29.1	0	0.0	29.513	12.0	4.0
338	2015-12-05	24.8	19.5	30.6	0.1	1.0	32.451	12.0	5.0
339	2015-12-06	23.12	20.6	28	0.1	1.0	32.780	12.0	6.0
340	2015-12-07	20.04	18	23.9	47.8	0.0	23.375	12.0	0.0

In [22]:
```
df1.set_index('Data',inplace=True)
```

Question-4: Handle null values

Tasks to do:

Only drop those instances where all values are null

Also, check the duplicate values

In [23]:
```
print(df1.isnull().sum())
print(df1.shape)
```

```
Temperatura Media (C)          576
Temperatura Minima (C)         576
Temperatura Maxima (C)         576
Precipitacao (mm)              576
Final de Semana                576
Consumo de cerveja (litros)    576
Month                          576
day                            576
dtype: int64
(941, 8)
```

In [24]:
```
print(df1.isnull().all(axis=1).sum()) # calculate the number of rows which have null values in all columns
```

```
576
```

In [25]:
```
df1.head()
```

Out[25]:

Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)	Month	day
2015-01-01	27.3	23.9	32.5	0	0.0	25.461	1.0	3.0
2015-01-02	27.02	24.5	33.5	0	0.0	28.972	1.0	4.0
2015-01-03	24.82	22.4	29.9	0	1.0	30.814	1.0	5.0
2015-01-04	23.98	21.5	28.6	1.2	1.0	29.799	1.0	6.0
2015-01-05	23.82	21	28.3	0	0.0	28.900	1.0	0.0

We can see that the 576 instances have all null values in all columns. So we can easily drop those instances

In [26]:
```
df1.dropna(how='all',inplace=True)
```

In [27]:
```
df1.shape
```

Out[27]: (365, 8)

In [28]:
```
print(df1.isnull().sum())
```

```
Temperatura Media (C)          0
Temperatura Minima (C)         0
Temperatura Maxima (C)         0
Precipitacao (mm)              0
Final de Semana                0
Consumo de cerveja (litros)    0
Month                          0
day                            0
dtype: int64
```

Now we don't have any null values

In [29]:
```
if df1.duplicated().any():
  print('True: duplicate instances')
else:
  print('False: No duplicate instances')
```

```
False: No duplicate instances
```

Question-5: Handle data types of each features

Tasks to do:

Check the data-types of the features

Convert them to appropriate data types

In [30]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 365 entries, 2015-01-01 to 2015-12-31
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Temperatura Media (C)        365 non-null    object 
 1   Temperatura Minima (C)       365 non-null    object 
 2   Temperatura Maxima (C)       365 non-null    object 
 3   Precipitacao (mm)            365 non-null    object 
 4   Final de Semana              365 non-null    float64
 5   Consumo de cerveja (litros)  365 non-null    float64
 6   Month                        365 non-null    float64
 7   day                          365 non-null    float64
dtypes: float64(4), object(4)
memory usage: 25.7+ KB
```

The columns with dtype object will be needed to be converted to appropriate dtype

In [31]:
```
df1['Temperatura Media (C)']=df1[['Temperatura Media (C)']].astype(float)
df1['Temperatura Minima (C)'] = df1['Temperatura Minima (C)'].astype(float)
df1['Temperatura Maxima (C)'] = df1['Temperatura Maxima (C)'].astype(float)
df1['Precipitacao (mm)'] = df1['Precipitacao (mm)'].astype(float)
# Final de semana is a categorical column(like yes or no) so it should be int, not float
df1['Final de Semana'] = df1['Final de Semana'].astype(int)
```

In [32]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 365 entries, 2015-01-01 to 2015-12-31
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Temperatura Media (C)        365 non-null    float64
 1   Temperatura Minima (C)       365 non-null    float64
 2   Temperatura Maxima (C)       365 non-null    float64
 3   Precipitacao (mm)            365 non-null    float64
 4   Final de Semana              365 non-null    int64  
 5   Consumo de cerveja (litros)  365 non-null    float64
 6   Month                        365 non-null    float64
 7   day                          365 non-null    float64
dtypes: float64(7), int64(1)
memory usage: 25.7 KB
```

In [33]:
```
df1.describe()
```

Out[33]:

	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)	Month	day
count	365.000000	365.000000	365.000000	365.000000	365.000000	365.000000	365.000000	365.0
mean	21.226356	17.461370	26.611507	5.196712	0.284932	25.401367	6.526027	3.0
std	3.180108	2.826185	4.317366	12.417844	0.452001	4.399143	3.452584	2.0
min	12.900000	10.600000	14.500000	0.000000	0.000000	14.343000	1.000000	0.0
25%	19.020000	15.300000	23.800000	0.000000	0.000000	22.008000	4.000000	1.0
50%	21.380000	17.900000	26.900000	0.000000	0.000000	24.867000	7.000000	3.0
75%	23.280000	19.600000	29.400000	3.200000	1.000000	28.631000	10.000000	5.0
max	28.860000	24.500000	36.500000	94.800000	1.000000	37.937000	12.000000	6.0

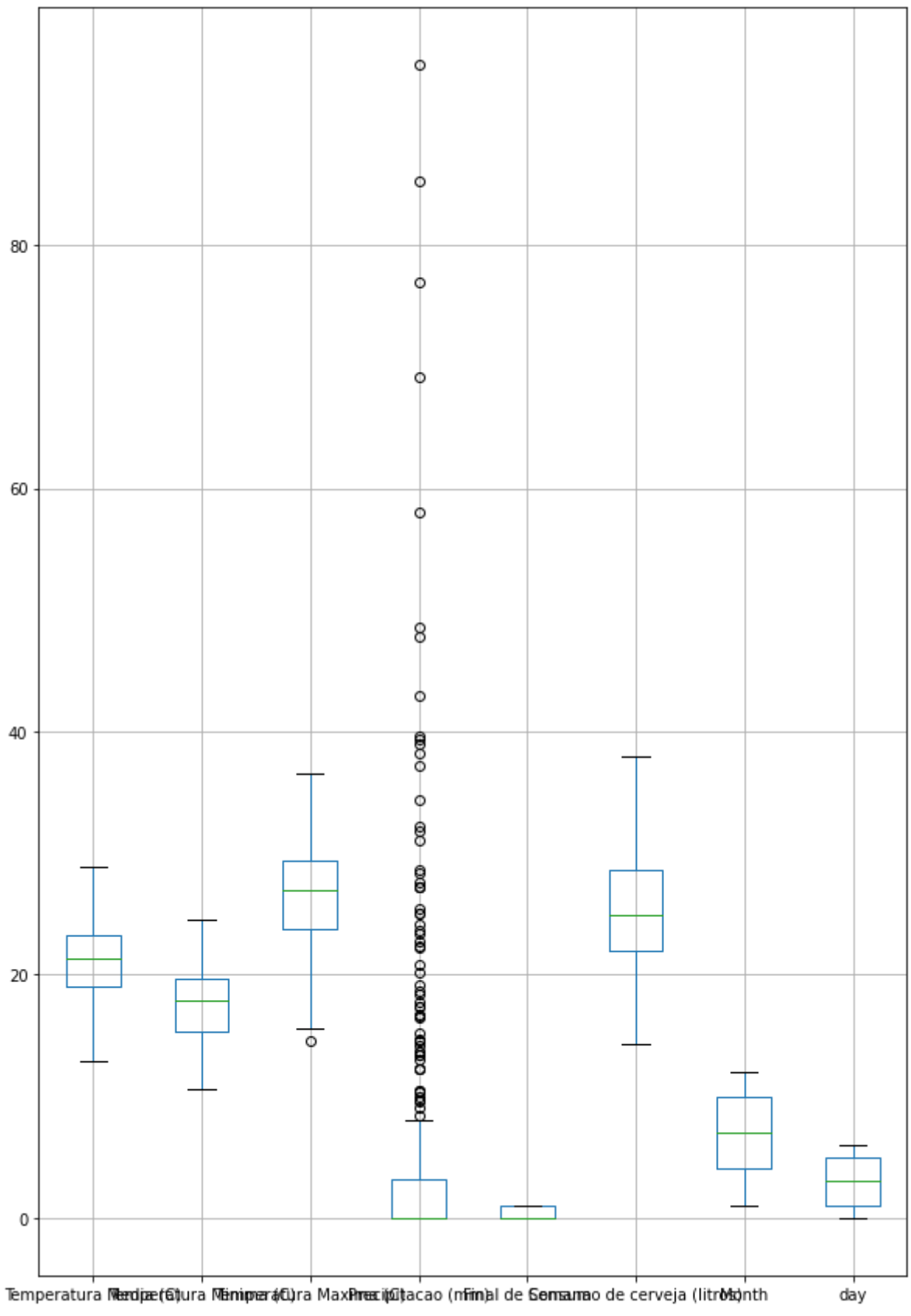
Question-6: Handle Outlier Data

Tasks to do:

Check for outlier in all the columns using boxplot

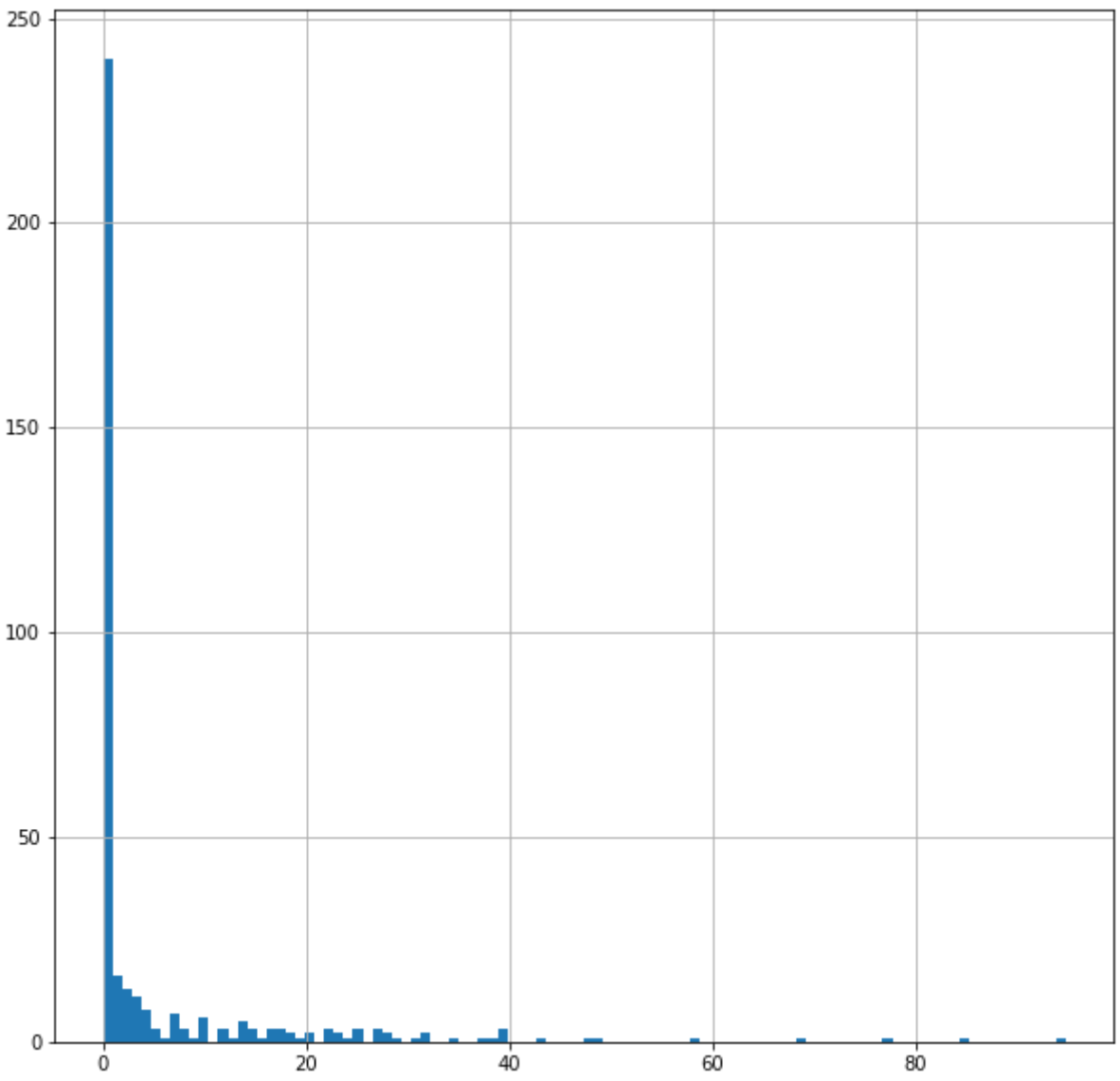
Analyze the column with outliers

In [34]:
```
import matplotlib.pyplot as plt
df1.boxplot(figsize=(10,15))
plt.show()
```



Column 'Precipitacio' seems to have lots of outlier. Let's try to understand this using the distribution of the data

In [35]:
```
df1['Precipitacao (mm)'].hist(bins=100,figsize=(10,10))
plt.show()
```



In [36]:
```
print(df1['Precipitacao (mm)'][df1['Precipitacao (mm)']==0].value_counts())
```

```
0.0    218
Name: Precipitacao (mm), dtype: int64
```

We can see out of 365, 218 values are 0

We can see how the data is mostly skewed thus having so many outliers. It can also be possible that the values with 0 percipitation are the instance where percipitation was not recorded. Lets clip all the values over 40 in column 'Precipitacao (mm)' to 40.

In [37]:
```
df1['Precipitacao (mm)'] = np.clip(df1['Precipitacao (mm)'], 0, 40)
```

Question-7: Calculate correlation and compare

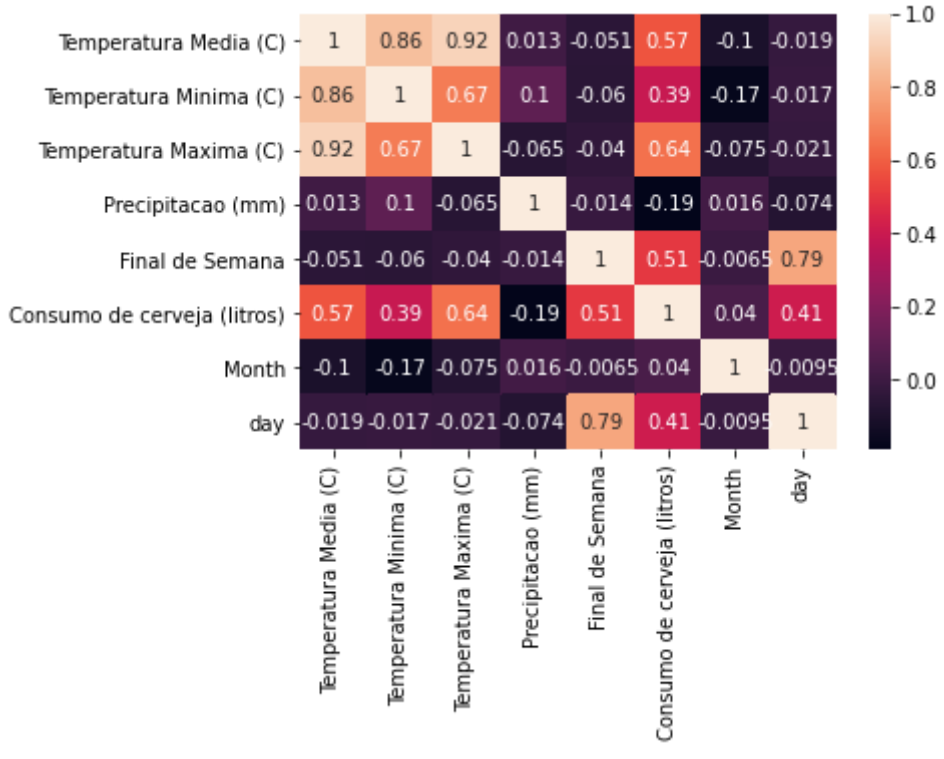
Tasks to do:

Plot the correlation between features

Analyze the correlation of independent features with respect to dependent features

In [38]:
```
import seaborn as sns
correlation = df1.corr()
sns.heatmap(correlation, annot = True)
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb524211630>



All the features are showing a high correlation with the output feature except the 'Month' feature. In the case of the 'Precipitacao (mm)' feature, which has lots of values as 0, still, it is showing quite good correlation, and hence we will keep it.

Question-8: Split the data into training and testing datasets

Tasks to do:

Split the dataset using sklearn, with 20% for testing with random_state=7

In [39]:
```
from sklearn.model_selection import train_test_split
X= df1.drop(columns=['Consumo de cerveja (litros)'],axis=1)
y= df1['Consumo de cerveja (litros)']
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.20,random_state = 7)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(292, 7)
(73, 7)
(292,)
(73,)
```

Question-9: Train the model

Tasks to do:

Train a linear regression model for prediction

Also, print the coefficients and intercept from the trained model

In [40]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
print('The final coefficients after training is:',lr.coef_)
print('The final intercept after training is:',lr.intercept_)
```

```
The final coefficients after training is: [-1.34341349e-01  1.22055186e-01  7.03305117e-01 -5.80658209e-02
  5.24988993e+00  1.36610989e-01  2.42604471e-03]
The final intercept after training is: 5.299119057068435
```

Question-10: Evaluate the model

Tasks to do:

Predict the consumption for the test data

Evaluate the model using R2 score

Evaluate the model using Mean Absolute Error

Evaluate the model using Root Mean Squared Error

In [41]:
```
from sklearn.metrics import r2_score,mean_squared_error, mean_absolute_error
y_pred = lr.predict(X_test)
print("r2 score of our model is:", r2_score(y_test,y_pred))
print("mean absolute error of our model is:", mean_absolute_error(y_test,y_pred))
print("root mean squared error of our model is:", mean_squared_error(y_test,y_pred,squared=False))
```

```
r2 score of our model is: 0.6692125883575043
mean absolute error of our model is: 2.0233917644808708
root mean squared error of our model is: 2.4737414676598464
```

Linear Regression

Linear regression is a technique that predicts the value of variable y based on the values of x

- Linear regression is a supervised algorithm
- Used for finding the relationship between the independent and dependent variable
- Find the relation between two or more continuous variables

Scenario 1: Salary Prediction

You have been provided the 'Salary' dataset. You have to predict the salary for a given period of experience of a person.

Dataset Description:

The dataset contains two features:

YearsExperience: Employee's years of experience

Salary: Salary earned by the employee

Tasks to be performed:

1. Load the data, check its shape and check for null values - Beginner
2. Split the dataset for training and testing - Beginner
3. Implement Linear regression without using sklearn - Advanced
4. Perform Prediction on test data - Intermediate
5. Evaluate the model - Beginner
6. Train the model using sklearn - Intermediate (Bridging Question)
7. Predict the salary on test data and evaluate the model - Intermediate (Bridging Question)

Topics Covered:

Training a **Simple Linear Regression** model **without using sklearn**

Training a **Simple Linear Regression** model **using sklearn**

Predicting using the trained model

Evaluating a model: **R2-score** and **Mean Squared Error**

Fetch and Download the data

In [ ]:
```
!wget https://www.dropbox.com/s/swy1ch89q8uz97m/Salary_Data.csv
```

```
--2020-07-15 03:24:40--  https://www.dropbox.com/s/swy1ch89q8uz97m/Salary_Data.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/swy1ch89q8uz97m/Salary_Data.csv [following]
--2020-07-15 03:24:41--  https://www.dropbox.com/s/raw/swy1ch89q8uz97m/Salary_Data.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc21035649a9a432b397ce790a4f.dl.dropboxusercontent.com/cd/0/inline/A7gWFS8CZKaHgO_v7NjAbZHBvw95lt0u
nPNGeJMVrbTSYiW2ZPHumN3XtzgO-WVA3P9Un96Ua9fPRwyfM4-l30rwvSkIeXnhl9aRdW7w1aLdQH6FDaQmdsblSEEuTw6P7vg/file# [following]
--2020-07-15 03:24:41--  https://uc21035649a9a432b397ce790a4f.dl.dropboxusercontent.com/cd/0/inline/A7gWFS8CZKaHgO_v7
NjAbZHBvw95lt0unPNGeJMVrbTSYiW2ZPHumN3XtzgO-WVA3P9Un96Ua9fPRwyfM4-l30rwvSkIeXnhl9aRdW7w1aLdQH6FDaQmdsblSEEuTw6P7vg/fi
le
Resolving uc21035649a9a432b397ce790a4f.dl.dropboxusercontent.com (uc21035649a9a432b397ce790a4f.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to uc21035649a9a432b397ce790a4f.dl.dropboxusercontent.com (uc21035649a9a432b397ce790a4f.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 454 [text/plain]
Saving to: ‘Salary_Data.csv’

Salary_Data.csv     100%[===================>]     454  --.-KB/s    in 0s

2020-07-15 03:24:41 (68.4 MB/s) - ‘Salary_Data.csv’ saved [454/454]
```

Question-1: Load the data in a pandas DataFrame. Check if the dataset contains any null values also check the shape of the dataset

In [ ]:
```
import pandas as pd
df=pd.read_csv('/content/Salary_Data.csv')
```

In [ ]:
```
print(df.head())
print(df.shape)
print(df.isnull().sum())
```

```
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
(30, 2)
YearsExperience    0
Salary             0
dtype: int64
```

There are no null values, so we can move forward for prediction

Question-2: Divide the dependent and independent features from the Dataframe, also divide the dataset in training and testing set, keeping 10% of the data for testing. Check the shape of the independent and dependent features.

In [ ]:
```
from sklearn.model_selection import train_test_split
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.10,random_state = 7)
print(X_train.shape)
print(X_test.shape)
```

```
(27, 1)
(3, 1)
```

Question-3: Implement Linear regression without using sklearn

In [ ]:
```
#This function will be used to find the value of y intercept and slope
def coef_estimation(x, y):
    # number of observations/points
    n = np.size(x)

    # Find mean of x and y
    x_mean = np.mean(x)
    y_mean = np.mean(y)

    # Use sum of square to find out the intercept and slope

    SS_xy = np.sum(y*x) - (n * y_mean * x_mean)
    SS_xx = np.sum(x*x) - (n * x_mean * x_mean)

    # calculating regression coefficients
    slope = SS_xy / SS_xx
    y_intercept = y_mean - slope*x_mean

    return(y_intercept, slope)
```

In [ ]:
```
#changing the shape of the input since our method coef_estimation() expects 1d array
import numpy as np
X_train1=np.ravel(X_train)
print(X_train1.shape)
X_test1=np.ravel(X_test)
print(X_test1.shape)
```

```
(27,)
(3,)
```

Question-4: Perform prediction on test data using our model

In [ ]:
```
b = coef_estimation(X_train1, y_train)
print("Estimated coefficients:\nc = {} \nm = {}".format(b[0], b[1]))

y_pred_scratch = b[1]*X_test1 + b[0]
```

```
Estimated coefficients:
c = 24525.043001661514
m = 9589.49001292227
```

Question-5: Evaluate the model on test data

In [ ]:
```
from sklearn.metrics import r2_score,mean_squared_error
print("r2 score of our model is:", r2_score(y_test,y_pred_scratch))
print("mean absolute error of our model is:", mean_squared_error(y_test,y_pred_scratch))
```

```
r2 score of our model is: 0.8743350638929301
mean absolute error of our model is: 48721996.58965842
```

Question-6: Using simple linear regression from sklearn, train the model for prediction

In [ ]:
```
from sklearn.linear_model import LinearRegression
regr = LinearRegression()
regr.fit(X_train,y_train)
```

Out[ ]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Question-7: Predict the salary on the testing data and evaluate the model using R2 score and mean square error

In [ ]:
```
from sklearn.metrics import r2_score,mean_squared_error
y_pred = regr.predict(X_test)
print('Predictions for test data:', y_pred[:5])
print("r2 score of our model is:", r2_score(y_test,y_pred))
print("mean absolute error of our model is:", mean_squared_error(y_test,y_pred))
```

```
Predictions for test data: [38909.27802104 75349.34007015 36991.38001846]
r2 score of our model is: 0.8743350638929295
mean absolute error of our model is: 48721996.589658655
```

We can see that R2 score and mean absolute error from our model and sklearn's model are very similar

Scenario 2: House price prediction

You are provided with the California housing dataset. Based on the given parameters of a house, predict its price.

Dataset Description:

The dataset contains nine features:

longitude: A measure of how far west a house is; a higher value is farther west

latitude: A measure of how far north a house is; a higher value is farther north

housingMedianAge: Median age of a house within a block; a lower number is a newer building

total rooms: Total number of rooms within a block

total bedrooms: Total number of bedrooms within a block

population: Total number of people residing within a block

households: Total number of households, a group of people residing within a home unit, for a block

median income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)

median house value: Median house value for households within a block (measured in US Dollars)

Tasks to be performed:

1. Load the data, check its shape and check for null values - Beginner
2. Split the dataset for training and testing - Beginner
3. Train the model using sklearn - Beginner (Bridging Question)
4. Predict the prices on test data and evaluate the model - Beginner (Bridging Question)
5. Find coefficient and intercept using the trained model - Beginner (Bridging Question)

Topics Covered:

Training a **Linear Regression** model

Predicting using the trained model

Evaluating a model: **R2-score** and **Mean Absolute Error**

Finding out **coefficients** and **intercept**

Fetch and download the data

In [ ]:
```
!wget https://www.dropbox.com/s/id6qa4ryx3q0qh1/california_housing_train.csv
```

```
--2020-07-15 03:29:26--  https://www.dropbox.com/s/id6qa4ryx3q0qh1/california_housing_train.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/id6qa4ryx3q0qh1/california_housing_train.csv [following]
--2020-07-15 03:29:26--  https://www.dropbox.com/s/raw/id6qa4ryx3q0qh1/california_housing_train.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc809fd68c939101a91d1293a1d8.dl.dropboxusercontent.com/cd/0/inline/A7idKpLQu06ZqBZi3i0vWixlCcY58ijk
gLI62G_oXng0fS3UczifGneisGm2dOhbaRUlBrlvP7FED8185e71E0F8WgqgmZlKyolaKcl3Oh2C31nvSlKDYtTqIO9vhOQXC9U/file# [following]
--2020-07-15 03:29:27--  https://uc809fd68c939101a91d1293a1d8.dl.dropboxusercontent.com/cd/0/inline/A7idKpLQu06ZqBZi3
i0vWixlCcY58ijkgLI62G_oXng0fS3UczifGneisGm2dOhbaRUlBrlvP7FED8185e71E0F8WgqgmZlKyolaKcl3Oh2C31nvSlKDYtTqIO9vhOQXC9U/fi
le
Resolving uc809fd68c939101a91d1293a1d8.dl.dropboxusercontent.com (uc809fd68c939101a91d1293a1d8.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to uc809fd68c939101a91d1293a1d8.dl.dropboxusercontent.com (uc809fd68c939101a91d1293a1d8.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1706430 (1.6M) [text/plain]
Saving to: ‘california_housing_train.csv’

california_housing_ 100%[===================>]   1.63M  9.84MB/s    in 0.2s

2020-07-15 03:29:28 (9.84 MB/s) - ‘california_housing_train.csv’ saved [1706430/1706430]
```

Question-1: Load the data in a pandas DataFrame. Check if the dataset contains any null values also check the shape of the dataset

In [ ]:
```
import pandas as pd
df = pd.read_csv('/content/sample_data/california_housing_train.csv')
print(df.isnull().sum())
print(df.shape)
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
(17000, 9)
```

There are no null values, so we can move forward for prediction

Question-2: Divide the dependent and independent features from the data frame, also divide the dataset into training and testing set, keeping the last 1000 instances for testing. Check the shape of the independent and dependent features

In [ ]:
```
X_train=df.iloc[:-1000,:-1]
y_train=df.iloc[:-1000,-1]
X_test = df.iloc[-1000:, :-1]
y_test = df.iloc[-1000:,-1]
print('Shape of X_train is:', X_train.shape)
print('Shape of y_train is:', y_train.shape)
print('Shape of X_test is:', X_test.shape)
print('Shape of y_test is:', y_test.shape)
```

```
Shape of X_train is: (16000, 8)
Shape of y_train is: (16000,)
Shape of X_test is: (1000, 8)
Shape of y_test is: (1000,)
```

Question-3: Apply linear regression to train a model for prediction

In [ ]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
```

Out[ ]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Question-4: Predict the prices from the test data and calculate r2 score and mean absolute error

In [ ]:
```
from sklearn.metrics import r2_score, mean_absolute_error
y_pred = lr.predict(X_test)
print('Predictions for test data:', y_pred[:5])
print("r2 score of our model is:", r2_score(y_test,y_pred))
print("mean absolute error of our model is:", mean_absolute_error(y_test,y_pred))
```

```
Predictions for test data: [147162.8573019  225302.65308351 152796.4573683  308870.26419818
 283483.75481794]
r2 score of our model is: 0.6777386734175752
mean absolute error of our model is: 53606.950938384194
```

Question-5: From the trained model find the coefficients and intercept

In [ ]:
```
print('The final coefficients after training is:',lr.coef_)
print('The final intercept after training is:',lr.intercept_)
```

```
The final coefficients after training is: [-4.20851753e+04 -4.20954131e+04  1.03051140e+03 -8.79478522e+00
  1.21229265e+02 -3.75749690e+01  3.92433254e+01  4.04064982e+04]
The final intercept after training is: -3520006.3058233857
```

Scenario 3: Medical Cost Prediction

You are provided with the medical cost dataset. You need to predict individual medical costs billed by health insurance.

Dataset Description:

The dataset contains seven features:

age: age of the primary beneficiary

sex: gender of primary beneficiary female, male

bmi: Body mass index, providing an understanding of the body, weights that are relatively high or low relative to height, an objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9

children: Number of children covered by health insurance / Number of dependents

smoker: Smokes or not

region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest

charges: Individual medical costs billed by health insurance

Tasks to be performed:

1. Load the data, check its shape and check for null values - Beginner
2. Convert categorical feature to numerical values - Intermediate (Bridging Question)
3. Split the dataset for training and testing - Beginner
4. Train the model using sklearn - Beginner (Bridging Question)
5. Find the intercept and coefficient from the trained model - Beginner
6. Predict the prices of test data and evaluate the model - Beginner (Bridging Question)

Topics Covered:

Training a **Linear Regression** model

Predicting using the trained model

Evaluating a model: **R2-score** and **Root Mean Squared Error**

Finding out **coefficients** and **intercept**

Fetch and download the data

In [ ]:
```
!wget https://www.dropbox.com/s/6ghwbs2o1r15zu4/insurance.csv
```

```
--2020-07-15 03:30:32--  https://www.dropbox.com/s/6ghwbs2o1r15zu4/insurance.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/6ghwbs2o1r15zu4/insurance.csv [following]
--2020-07-15 03:30:32--  https://www.dropbox.com/s/raw/6ghwbs2o1r15zu4/insurance.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc5024eab2f99b0ca01ba23e2c6b.dl.dropboxusercontent.com/cd/0/inline/A7ghPi-PKfkeSk5K_JsbfzSgcog5dmBe
JbsnmaE3-TI6RvlDm8OXVnaZr82nujdCCGGlJPtpSwsw8p9A1gQjOJ84YdwVcii7NL2pHqKCEpjMK2cmM8m5x4i_VzmFC5qoK6c/file# [following]
--2020-07-15 03:30:32--  https://uc5024eab2f99b0ca01ba23e2c6b.dl.dropboxusercontent.com/cd/0/inline/A7ghPi-PKfkeSk5K_
JsbfzSgcog5dmBeJbsnmaE3-TI6RvlDm8OXVnaZr82nujdCCGGlJPtpSwsw8p9A1gQjOJ84YdwVcii7NL2pHqKCEpjMK2cmM8m5x4i_VzmFC5qoK6c/fi
le
Resolving uc5024eab2f99b0ca01ba23e2c6b.dl.dropboxusercontent.com (uc5024eab2f99b0ca01ba23e2c6b.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to uc5024eab2f99b0ca01ba23e2c6b.dl.dropboxusercontent.com (uc5024eab2f99b0ca01ba23e2c6b.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55628 (54K) [text/plain]
Saving to: ‘insurance.csv’

insurance.csv       100%[===================>]  54.32K  --.-KB/s    in 0.03s

2020-07-15 03:30:33 (1.72 MB/s) - ‘insurance.csv’ saved [55628/55628]
```

Question-1: Load the data in a pandas DataFrame. Check the shape of the dataset, also check if the dataset contains any null values

In [ ]:
```
df=pd.read_csv('/content/insurance.csv')
```

In [ ]:
```
print(df.head())
df.shape
```

```
   age     sex     bmi  children smoker     region      charges
0   19  female  27.900         0    yes  southwest  16884.92400
1   18    male  33.770         1     no  southeast   1725.55230
2   28    male  33.000         3     no  southeast   4449.46200
3   33    male  22.705         0     no  northwest  21984.47061
4   32    male  28.880         0     no  northwest   3866.85520
```

Out[ ]:
```
(1338, 7)
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

There are no null values, so we can move forward for prediction

Question-2: We can not use categorical features for prediction so convert categorical features to numeric values using One Hot Encoding

In [ ]:
```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
d=df.select_dtypes(include=[object])
encoded_labels = ohe.fit_transform(d).toarray()
```

In [ ]:
```
df_encoded=pd.DataFrame(encoded_labels)
df.drop(columns=['sex','smoker','region'],inplace=True)
df3 = pd.concat([df_encoded,df], ignore_index=True,axis=1)
```

In [ ]:
```
df3
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	19	27.900	0	16884.92400
1	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	18	33.770	1	1725.55230
2	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	28	33.000	3	4449.46200
3	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	33	22.705	0	21984.47061
4	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	32	28.880	0	3866.85520
...	...	...	...	...	...	...	...	...	...	...	...	...
1333	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	50	30.970	3	10600.54830
1334	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	18	31.920	0	2205.98080
1335	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	18	36.850	0	1629.83350
1336	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	21	25.800	0	2007.94500
1337	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	61	29.070	0	29141.36030

1338 rows × 12 columns

Question-3: Split the dependent and independent features. Split the dataset into training and testing features keeping 25% of the data for testing

In [ ]:
```
X=df3.iloc[:,:-1]
y=df3.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state = 7)
print(X_train.shape)
print(X_test.shape)
```

```
(1003, 11)
(335, 11)
```

Question-4: Apply linear regression to train a model for prediction

In [ ]:
```
regr = LinearRegression()
regr.fit(X_train,y_train)
```

Out[ ]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Question-5: Find the coefficients and intercept from the trained model

In [ ]:
```
print('The final coefficients after training is:',lr.coef_)
print('The final intercept after training is:',lr.intercept_)
```

```
The final coefficients after training is: [-4.20851753e+04 -4.20954131e+04  1.03051140e+03 -8.79478522e+00
  1.21229265e+02 -3.75749690e+01  3.92433254e+01  4.04064982e+04]
The final intercept after training is: -3520006.3058233857
```

Question-6: Predict the prices from the test data and calculate r2 score and root mean squared error

In [ ]:
```
from sklearn.metrics import r2_score,mean_squared_error
y_pred = regr.predict(X_test)
print('Predictions for test data:', y_pred[:5])
print("r2 score of our model is:", r2_score(y_test,y_pred))
print("root mean squared error of our model is:", mean_squared_error(y_test,y_pred,squared=False))
```

```
Predictions for test data: [15248.874306   11126.97945225 -2048.68105088 29282.63519248
  9070.8295246 ]
r2 score of our model is: 0.7509741262661105
root mean squared error of our model is: 6080.97796761699
```

Scenario 4: Startup's Profit Prediction

You are provided with the '50_Startups' data. Using the given features, you have to predict the profit of these startups.

Dataset Description:

The dataset contains 5 features:

R&D Spend: Expenditures in Research and Development

Administration: Expenditures in Administration

Marketing Spend: Expenditures in Marketing

State: In which state the company belongs to

Profit: The profit made by the company

Tasks to be performed:

1. Load the data, check its shape and check for null values - Beginner
2. Convert categorical feature to numerical values - Intermediate
3. Split the dataset for training and testing - Beginner
4. Train the model using sklearn, also find the intercept and coefficient from the trained model - Beginner
5. Predict the profits of test data and evaluate the model - Beginner
6. Regularize the model using Ridge Regression and find the Score - Intermediate (Bridging Question)
7. Regularize the model using Lasso Regression and find the Score - Intermediate (Bridging Question)

Topics Covered:

Training a **Linear Regression** model

Predicting using the trained model

Evaluating a model: **R2-score** and **Root Mean Squared Error**

Finding out **coefficients** and **intercept**

Ridge Regression

Lasso Regression

Fetch and download the data

In [ ]:
```
!wget https://www.dropbox.com/s/zcebxexe0a4atiy/50_Startups.csv
```

```
--2020-07-15 03:20:16--  https://www.dropbox.com/s/zcebxexe0a4atiy/50_Startups.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/zcebxexe0a4atiy/50_Startups.csv [following]
--2020-07-15 03:20:16--  https://www.dropbox.com/s/raw/zcebxexe0a4atiy/50_Startups.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc513e8c58db658383ae9ff3edb2.dl.dropboxusercontent.com/cd/0/inline/A7j9LkhMssyrhO9ldScIaxDFtlTzIYpX
XZjUQ_4y5N77VUUCol20kGxwKOvKa6Fc5COsRN08n1G3hhYg6b9jMbznJXCwFI_TDxpgpkCeAkWOns5waLHgIWSPKE1cM6dEKmU/file# [following]
--2020-07-15 03:20:17--  https://uc513e8c58db658383ae9ff3edb2.dl.dropboxusercontent.com/cd/0/inline/A7j9LkhMssyrhO9ld
ScIaxDFtlTzIYpXXZjUQ_4y5N77VUUCol20kGxwKOvKa6Fc5COsRN08n1G3hhYg6b9jMbznJXCwFI_TDxpgpkCeAkWOns5waLHgIWSPKE1cM6dEKmU/fi
le
Resolving uc513e8c58db658383ae9ff3edb2.dl.dropboxusercontent.com (uc513e8c58db658383ae9ff3edb2.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to uc513e8c58db658383ae9ff3edb2.dl.dropboxusercontent.com (uc513e8c58db658383ae9ff3edb2.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2436 (2.4K) [text/plain]
Saving to: ‘50_Startups.csv’

50_Startups.csv     100%[===================>]   2.38K  --.-KB/s    in 0s

2020-07-15 03:20:17 (295 MB/s) - ‘50_Startups.csv’ saved [2436/2436]
```

Question-1: Load the data in a DataFrame, check if there are any null values. Check the shape of the data

In [ ]:
```
import pandas as pd
df = pd.read_csv('/content/50_Startups.csv')
print(df.isnull().sum())
print(df.head())
print(df.shape)
```

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
(50, 5)
```

There are no null values, so we can move forward for prediction

Question-2: We cannot use string objects for prediction. So, convert the categorical features to numerical values using Label Encoder

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['State']=le.fit_transform(df['State'])
df.head()
```

Out[ ]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	2	192261.83
1	162597.70	151377.59	443898.53	0	191792.06
2	153441.51	101145.55	407934.54	1	191050.39
3	144372.41	118671.85	383199.62	2	182901.99
4	142107.34	91391.77	366168.42	1	166187.94

Now we can see the name of states are converted into numerical labels

Question-3: Split the data into training and testing datasets, with 20% for testing with random_state=1.

In [ ]:
```
from sklearn.model_selection import train_test_split
X= df.iloc[:,:-1]
y= df.iloc[:,-1]
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.20,random_state = 1)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(40, 4)
(10, 4)
(40,)
(10,)
```

Question-4: Train a linear regression model for prediction. Also print the coefficients and intercept from the trained model

In [ ]:
```
lr = LinearRegression()
lr.fit(X_train,y_train)
print('The final coefficients after training is:',lr.coef_)
print('The final intercept after training is:',lr.intercept_)
```

```
The final coefficients after training is: [ 7.73632191e-01 -9.41863371e-03  2.93451013e-02  1.73159640e+02]
The final intercept after training is: 49637.34362243092
```

Question-5: Predict the profits for the test data, and evaluate the model

In [ ]:
```
from sklearn.metrics import r2_score,mean_squared_error
y_pred = lr.predict(X_test)
print("r2 score of our model is:", r2_score(y_test,y_pred))
print("mean absolute error of our model is:", mean_squared_error(y_test,y_pred,squared=False))
```

```
r2 score of our model is: 0.9649827631091706
mean absolute error of our model is: 8913.354557987737
```

Question-6: Regularize the model using Ridge Regression and find the Score

In [ ]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

In [ ]:
```
ridgeReg = Ridge(alpha=0.0005, normalize=True)

ridgeReg.fit(X_train,y_train)
pred = ridgeReg.predict(X_test)
score = ridgeReg.score(X_test,y_test)
score
```

Out[ ]:
```
0.9648637587691529
```

Question-7: Regularize the model using Lasso Regression and find the Score

In [ ]:
```
lassoReg = Lasso(alpha=0.03)
lassoReg.fit(X_train,y_train)
pred = lassoReg.predict(X_test)
score=lassoReg.score(X_test,y_test)
score
```

Out[ ]:
```
0.9649828004576062
```

In [ ]:
```

```

Classification

Classification is the process of grouping things according to similar features they share.

The outcome of classification is categorical, as opposed to a regression where the outcome is continuous.

Logistic Regression

The Statistical method used for classification aims to predict the probability of a dependent binary variable over one or more dependent variables.

Decision Tree

- It uses a graphical representation of all the possible solutions to a decision
- Decisions are mainly based on some conditions

Random Forest

Random Forest is an ensemble method made of decision trees.

Scenario 1: Heart Disease Prediction Using Logistic Regression

Industry: Healthcare

World Health Organization reported that there are 12 million deaths worldwide each year due to heart disease. Half the deaths are caused by cardiovascular diseases in the United States and other developing countries. The early prognosis of cardiovascular disorders will help make decisions about improvements in lifestyle in high-risk patients and, in effect, reduce the complications. This research aims to classify the most relevant / risk factors for cardiac disease and estimate the overall risk using logistic regression.

Dataset Description:

The dataset is publicly available on the Kaggle website (https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset) it is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD). The dataset provides the patients' information. It includes over 4,000 records and 15 attributes.

Variables/Attributes :

Each attribute is a potential risk factor. There are demographic, behavioral, and medical risk factors.

- Demographic: sex: male or female;(Nominal)
  - age: age of the patient;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)
- Behavioral
  - current smoker: whether or not the patient is a current smoker (Nominal)
  - cigsPerDay: the number of cigarettes that the person smoked on average in one day. (can be considered continuous as one can have any number of cigarettes, even half a cigarette.)
- Medical(history):
  - BPMeds: whether or not the patient was on blood pressure medication (Nominal)
  - prevalent stroke: whether or not the patient had previously had a stroke (Nominal)
  - prevalentHyp: whether or not the patient was hypertensive (Nominal)
  - diabetes: whether or not the patient had diabetes (Nominal)
- Medical(current):
  - totChol: total cholesterol level (Continuous)
  - sysBP: systolic blood pressure (Continuous)
  - diaBP: diastolic blood pressure (Continuous)
  - BMI: Body Mass Index (Continuous)
  - heartRate: heart rate (Continuous - In medical research, variables such as heart rate though, discrete, yet are considered continuous because of the large number of possible values.)
  - glucose: glucose level (Continuous)

Predict variable (desired target):

- Ten-year risk of coronary heart disease CHD (binary: "1", means "Yes," "0" means "No")

Tasks to be Performed

- Import the required libraries- Beginner
- Prepare the data (Data Processing)- Intermediate
- Check for missing value- Beginner
- Explore the data using EDA- Beginner
- Training a logistic regression model- Intermediate (Bridging Question)
- Evaluating the model- Advance (Bridging Question)

Topics Covered

- Logistic Regression

Question 1: Import the required libraries

In [ ]:
```
#Importing the Required Libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import confusion_matrix
import matplotlib.mlab as mlab
%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
ated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Question 2: Data Preprocessing

In [ ]:
```
!wget -O framingham.csv https://www.dropbox.com/s/83lq09bnj62zc3t/framingham.csv?dl=0
```

```
--2020-07-20 07:21:01--  https://www.dropbox.com/s/83lq09bnj62zc3t/framingham.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/83lq09bnj62zc3t/framingham.csv [following]
--2020-07-20 07:21:02--  https://www.dropbox.com/s/raw/83lq09bnj62zc3t/framingham.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucc8e12f0e1de9b484273213c6bd.dl.dropboxusercontent.com/cd/0/inline/A70tmYbzoAnCACSqeqgD2rKxLLnrMRhA
iEWZ1XuyewASNRCx_el_20-D3RuhYUMPxIIGY3vYgYLkzwZHs_dfsdOTiUhoV-iF37_RHcXqztDgIkRuFQHRxU8OEY3K0hrM94g/file# [following]
--2020-07-20 07:21:02--  https://ucc8e12f0e1de9b484273213c6bd.dl.dropboxusercontent.com/cd/0/inline/A70tmYbzoAnCACSqe
qgD2rKxLLnrMRhAiEWZ1XuyewASNRCx_el_20-D3RuhYUMPxIIGY3vYgYLkzwZHs_dfsdOTiUhoV-iF37_RHcXqztDgIkRuFQHRxU8OEY3K0hrM94g/fi
le
Resolving ucc8e12f0e1de9b484273213c6bd.dl.dropboxusercontent.com (ucc8e12f0e1de9b484273213c6bd.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to ucc8e12f0e1de9b484273213c6bd.dl.dropboxusercontent.com (ucc8e12f0e1de9b484273213c6bd.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 191803 (187K) [text/plain]
Saving to: ‘framingham.csv’

framingham.csv      100%[===================>] 187.31K  --.-KB/s    in 0.08s

2020-07-20 07:21:03 (2.23 MB/s) - ‘framingham.csv’ saved [191803/191803]
```

In [ ]:
```
heart_df=pd.read_csv("/content/framingham.csv")
heart_df.drop(['education'],axis=1,inplace=True)
heart_df.head()
```

Out[ ]:

	male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	T
0	1	39	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	
1	0	46	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	
2	1	48	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	
3	0	61	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	
4	0	46	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	

In [ ]:
```
heart_df.rename(columns={'male':'Sex_male'},inplace=True)
```

Question 3: Finding missing values

In [ ]:
```
heart_df.isnull().sum()
```

Out[ ]:
```
Sex_male            0
age                 0
currentSmoker       0
cigsPerDay         29
BPMeds             53
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            50
sysBP               0
diaBP               0
BMI                19
heartRate           1
glucose           388
TenYearCHD          0
dtype: int64
```

In [ ]:
```
count=0
for i in heart_df.isnull().sum(axis=1):
    if i>0:
        count=count+1
print('Total number of rows with missing values is ', count)
print('since it is only',round((count/len(heart_df.index))*100), 'percent of the entire dataset the rows with missing values are excluded.')
```

```
Total number of rows with missing values is  489
since it is only 12 percent of the entire dataset the rows with missing values are excluded.
```

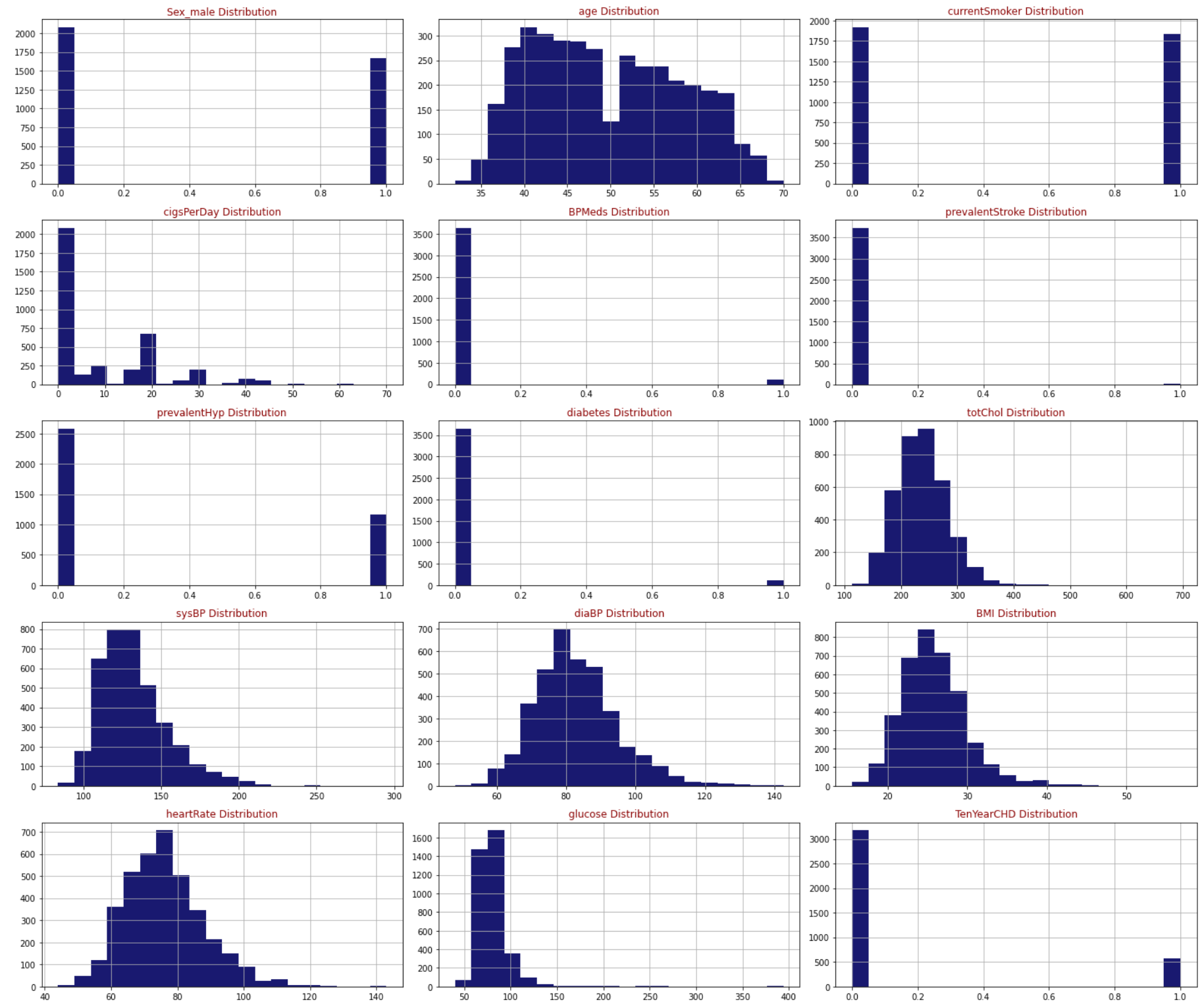
In [ ]:
```
#Removing/dropping the missing values
heart_df.dropna(axis=0,inplace=True)
```

In [ ]:
```
heart_df.isnull().sum()
```

Out[ ]:
```
Sex_male          0
age               0
currentSmoker     0
cigsPerDay        0
BPMeds            0
prevalentStroke   0
prevalentHyp      0
diabetes          0
totChol           0
sysBP             0
diaBP             0
BMI               0
heartRate         0
glucose           0
TenYearCHD        0
dtype: int64
```

Question 4: Explore the data using EDA

In [ ]:
```
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor='midnightblue')
        ax.set_title(feature+" Distribution",color='DarkRed')

    fig.tight_layout()
    plt.show()
draw_histograms(heart_df,heart_df.columns,6,3)
```

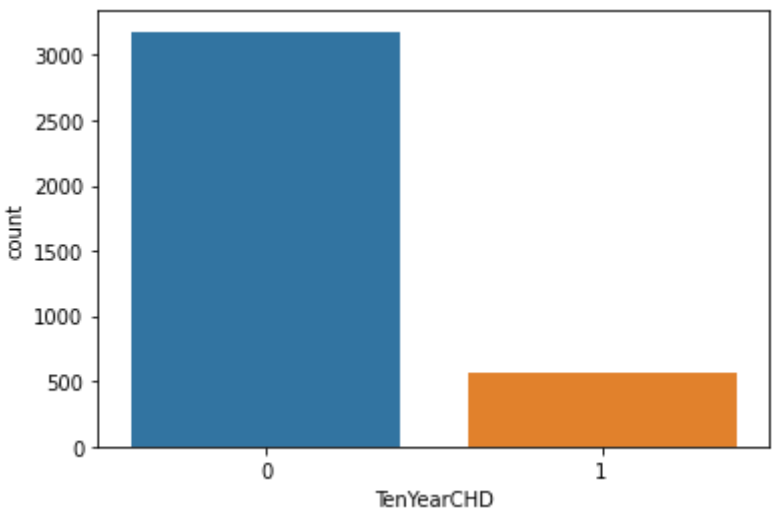


In [ ]:
```
heart_df.TenYearCHD.value_counts()
```

Out[ ]:
```
0    3179
1     572
Name: TenYearCHD, dtype: int64
```

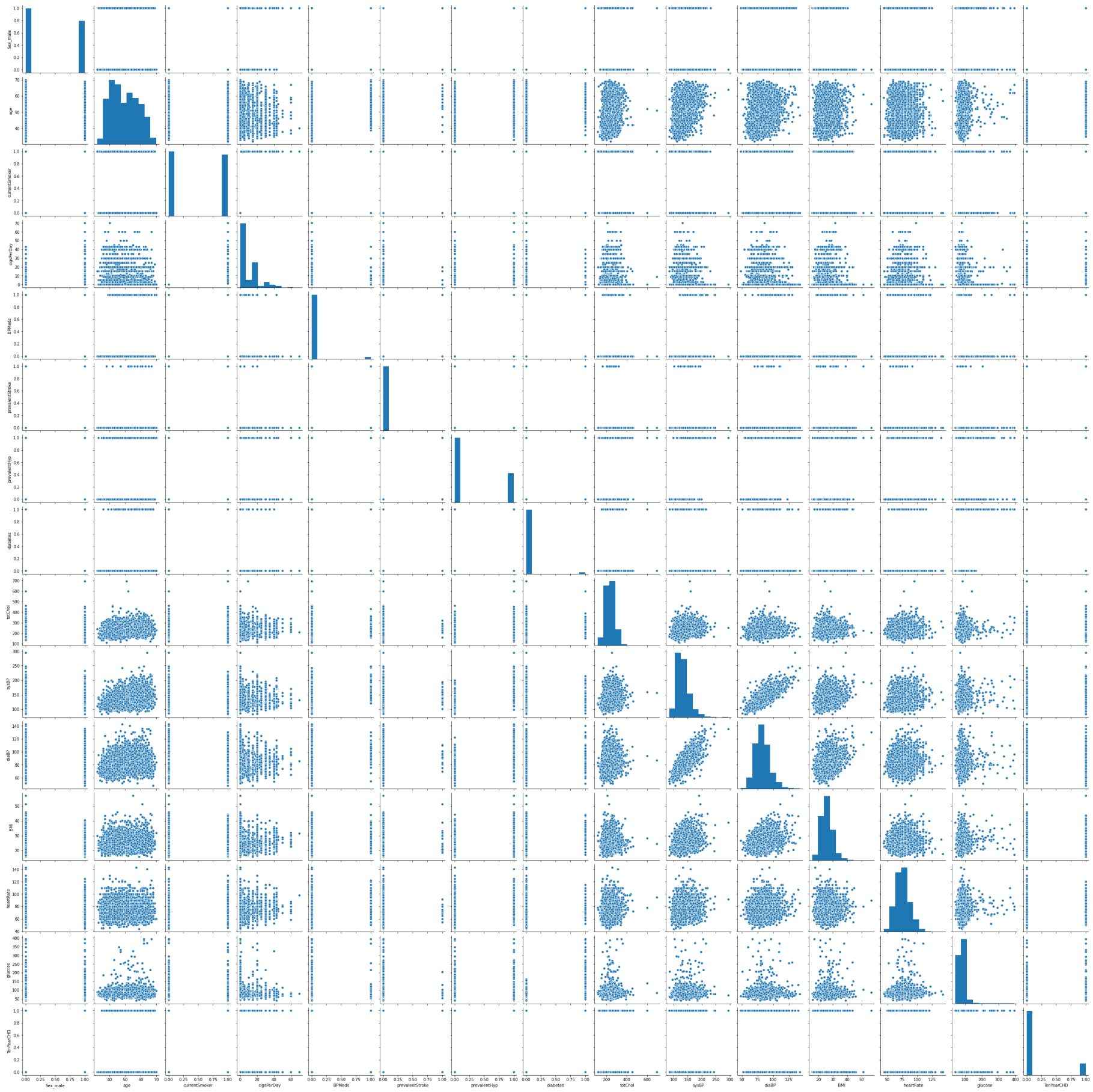
In [ ]:
```
sn.countplot(x='TenYearCHD',data=heart_df)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8110de3c8>
```



In [ ]:
```
sn.pairplot(data=heart_df)
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x7ff81110c6a0>



In [ ]:
```
heart_df.describe()
```

Out[ ]:

	Sex_male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP
count	3751.000000	3751.000000	3751.000000	3751.000000	3751.000000	3751.000000	3751.000000	3751.000000	3751.000000	3751.000000
mean	0.445215	49.573447	0.488403	9.008531	0.030392	0.005599	0.311917	0.027193	236.928019	132.368435
std	0.497056	8.570204	0.499932	11.925097	0.171686	0.074623	0.463338	0.162666	44.611594	22.046522
min	0.000000	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	113.000000	83.500000
25%	0.000000	42.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000	117.000000
50%	0.000000	49.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000	128.000000
75%	1.000000	56.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	264.000000	144.000000
max	1.000000	70.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696.000000	295.000000

Question 5: Building a Logistic Regression model

Logistic regression is a form of regression analysis used in statistics to predict the outcome of a categorical variable based on a set of predictors or independent variables. The dependent variable in logistic regression is always binary. Logistic regression is used mainly for estimation and also for estimating the probability of success.

In [ ]:
```
from statsmodels.tools import add_constant as add_constant
heart_df_constant = add_constant(heart_df)
heart_df_constant.head()
```

Out[ ]:

	const	Sex_male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate
0	1.0	1	39	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0
1	1.0	0	46	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0
2	1.0	1	48	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0
3	1.0	0	61	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0
4	1.0	0	46	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0

In [ ]:
```
st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
cols=heart_df_constant.columns[:-1]
model=sm.Logit(heart_df.TenYearCHD,heart_df_constant[cols])
result=model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.377036
         Iterations 7
```

Out[ ]:

Logit Regression Results

Dep. Variable:	TenYearCHD	**No. Observations:**	3751
Model:	Logit	**Df Residuals:**	3736
Method:	MLE	**Df Model:**	14
Date:	Mon, 20 Jul 2020	**Pseudo R-squ.:**	0.1170
Time:	07:22:22	**Log-Likelihood:**	-1414.3
converged:	True	**LL-Null:**	-1601.7
Covariance Type:	nonrobust	**LLR p-value:**	2.439e-71

	coef	std err	z	P>|z|	[0.025	0.975]
const	-8.6532	0.687	-12.589	0.000	-10.000	-7.306
Sex_male	0.5742	0.107	5.345	0.000	0.364	0.785
age	0.0641	0.007	9.799	0.000	0.051	0.077
currentSmoker	0.0739	0.155	0.478	0.633	-0.229	0.377
cigsPerDay	0.0184	0.006	3.000	0.003	0.006	0.030
BPMeds	0.1448	0.232	0.623	0.533	-0.310	0.600
prevalentStroke	0.7193	0.489	1.471	0.141	-0.239	1.678
prevalentHyp	0.2142	0.136	1.571	0.116	-0.053	0.481
diabetes	0.0022	0.312	0.007	0.994	-0.610	0.614
totChol	0.0023	0.001	2.081	0.037	0.000	0.004
sysBP	0.0154	0.004	4.082	0.000	0.008	0.023
diaBP	-0.0040	0.006	-0.623	0.533	-0.016	0.009
BMI	0.0103	0.013	0.827	0.408	-0.014	0.035
heartRate	-0.0023	0.004	-0.549	0.583	-0.010	0.006
glucose	0.0076	0.002	3.409	0.001	0.003	0.012

The results above show some of the attributes with P-value higher than the preferred alpha(5%) and thereby showing a low statistically significant relationship with the probability of heart disease. The backward elimination approach is used here to remove those attributes with the highest P-value one at a time, followed by repeatedly running the regression until all attributes have P Values less than 0.05.

Feature Selection: Backward elemination (P-value approach)

In [ ]:
```
def back_feature_elem (data_frame,dep_var,col_list):
    """ Takes in the dataframe, the dependent variable and a list of column names, runs the regression repeatedly elem
inating feature with the highest
    P-value above alpha one at a time and returns the regression summary with all p-values below alpha"""

    while len(col_list)>0 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)

result=back_feature_elem(heart_df_constant,heart_df.TenYearCHD,cols)
```

In [ ]:
```
result.summary()
```

Out[ ]:

Logit Regression Results

Dep. Variable:	TenYearCHD	**No. Observations:**	3751
Model:	Logit	**Df Residuals:**	3744
Method:	MLE	**Df Model:**	6
Date:	Mon, 20 Jul 2020	**Pseudo R-squ.:**	0.1149
Time:	07:22:22	**Log-Likelihood:**	-1417.7
converged:	True	**LL-Null:**	-1601.7
Covariance Type:	nonrobust	**LLR p-value:**	2.127e-76

	coef	std err	z	P>|z|	[0.025	0.975]
const	-9.1264	0.468	-19.504	0.000	-10.043	-8.209
Sex_male	0.5815	0.105	5.524	0.000	0.375	0.788
age	0.0655	0.006	10.343	0.000	0.053	0.078
cigsPerDay	0.0197	0.004	4.805	0.000	0.012	0.028
totChol	0.0023	0.001	2.106	0.035	0.000	0.004
sysBP	0.0174	0.002	8.162	0.000	0.013	0.022
glucose	0.0076	0.002	4.574	0.000	0.004	0.011

Logistic regression equation

$$P=\hspace{.2cm}e^{\beta_0 + \beta_1 X_1}\hspace{.2cm}/\hspace{.2cm}1+e^{\beta_0 +\beta_1 X_1}$$

When all features plugged in:

$$logit(p) = log(p/(1-p))=\beta_0 +\beta_1\hspace{.1cm} *\hspace{.2cm} Sexmale\hspace{.2cm}+\beta_2\hspace{.1cm} *
\hspace{.1cm}age\hspace{.2cm}+\hspace{.2cm}\beta_3\hspace{.1cm} *\hspace{.1cm} cigsPerDay\hspace{.2cm}+\hspace{.2cm}\beta_4
\hspace{.1cm}*\hspace{.1cm} totChol\hspace{.2cm}+\hspace{.2cm}\beta_5\hspace{.1cm} *\hspace{.1cm}
sysBP\hspace{.2cm}+\hspace{.2cm}\beta_6\hspace{.1cm} *\hspace{.1cm} glucose\hspace{.2cm}$$

Interpreting the results: Odds Ratio, Confidence Intervals and Pvalues

In [ ]:
```
params = np.exp(result.params)
conf = np.exp(result.conf_int())
conf['OR'] = params
pvalue=round(result.pvalues,3)
conf['pvalue']=pvalue
conf.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf))
```

```
            CI 95%(2.5%)  CI 95%(97.5%)  Odds Ratio  pvalue
const           0.000043       0.000272    0.000109   0.000
Sex_male        1.455242       2.198536    1.788687   0.000
age             1.054483       1.080969    1.067644   0.000
cigsPerDay      1.011733       1.028128    1.019897   0.000
totChol         1.000158       1.004394    1.002273   0.035
sysBP           1.013292       1.021784    1.017529   0.000
glucose         1.004346       1.010898    1.007617   0.000
```

- This fitted model shows that holding all other features constant, the odds of getting diagnosed with heart disease for males (sex_male = 1) over that of females (sex_male = 0) is exp(0.5815) = 1.788687. In terms of percent change, we can say that males' odds are 78.8% higher than the odds for females.
- The coefficient for age says that holding all others constant, and we will see a 7% increase in the odds of getting diagnosed with CDH for a one year increase in age since exp(0.0655) = 1.067644.
- Similarly, with every extra cigarette one smokes, there is a 2% increase in the odds of CDH.
- For the total cholesterol level and glucose level, there is no significant change.
- There is a 1.7% increase in odds for every unit increase in systolic Blood Pressure.

Question 5: Training the model

Splitting into Train and Test

In [ ]:
```
import sklearn
new_features=heart_df[['age','Sex_male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD']]
x=new_features.iloc[:,:-1]
y=new_features.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=5)
```

Fit the data and Train the model

In [ ]:
```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Question 6: Model Evaluation

Model Accuracy

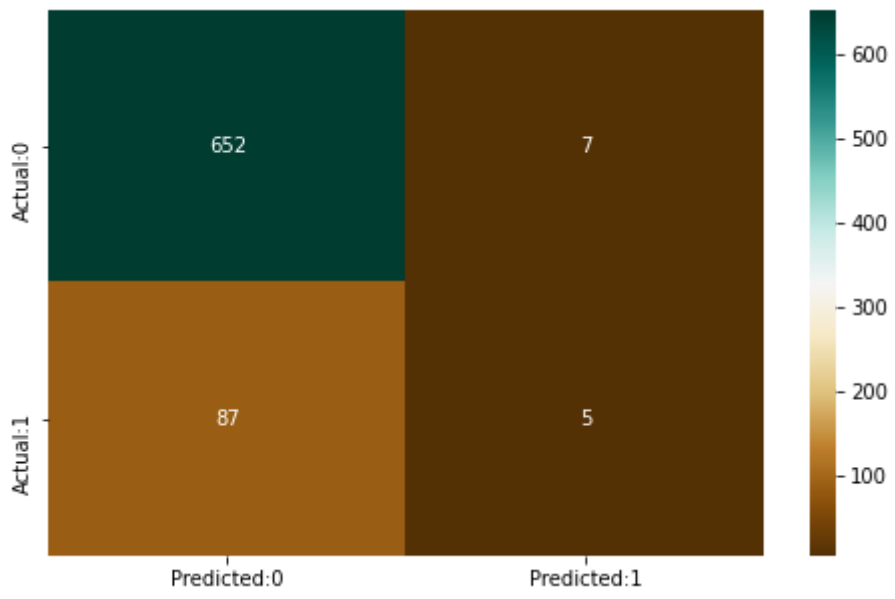
In [ ]:
```
sklearn.metrics.accuracy_score(y_test,y_pred)
```

Out[ ]:
```
0.8748335552596538
```

Confusion Matrix

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sn.heatmap(conf_matrix, annot=True,fmt='d',cmap="BrBG")
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff808348550>



The confusion matrix shows 652+5 = 657 correct predictions and 87+7= 94 incorrect ones.

True Positives: 5

True Negatives: 652

False Positives: 7 (*Type I error*)

False Negatives: 87 (*Type II error*)

In [ ]:
```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 652
True Positive 5
False Negative 87
False Positive 7
Sensitivity 0.05434782608695652
Specificity 0.9893778452200304
```

In [ ]:
```
print('The accuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n\n',

'Misclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)
```

```
The accuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.8748335552596538

 Misclassification = 1-Accuracy =  0.12516644474034622

 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.05434782608695652

 Specificity or True Negative Rate = TN/(TN+FP) =  0.9893778452200304

 Positive Predictive value = TP/(TP+FP) =  0.4166666666666667

 Negative predictive Value = TN/(TN+FN) =  0.8822733423545331

 Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  5.116459627329198

 Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.9558048813016804
```

From the above statistics, it is clear that the model is highly specific than sensitive. The negative values are predicted more accurately than the positives.

Predicted probabilities of 0 (Coronary Heart Disease: No) and 1 (Coronary Heart Disease: Yes) for the test data with a default classification threshold of 0.5

In [ ]:
```
y_pred_prob=logreg.predict_proba(x_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart disease (0)','Prob of Heart Disease (1)'])
y_pred_prob_df.head()
```

Out[ ]:

	Prob of no heart disease (0)	Prob of Heart Disease (1)
0	0.875026	0.124974
1	0.956183	0.043817
2	0.783473	0.216527
3	0.806581	0.193419
4	0.892879	0.107121

Lower the threshold

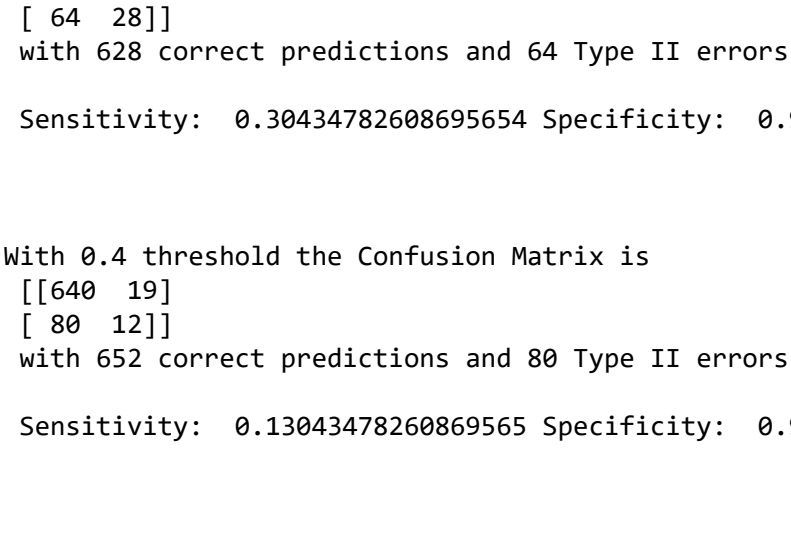
Since the model is predicting Heart disease, too many type II errors are not advisable. A False Negative (ignoring the probability of disease when there is one) is more dangerous than a False Positive in this case. Hence to increase the sensitivity, the threshold can be lowered.

In [ ]:
```
from sklearn.preprocessing import binarize
for i in range(1,5):
    cm2=0
    y_pred_prob_yes=logreg.predict_proba(x_test)
    y_pred2=binarize(y_pred_prob_yes,i/10)[:,1]
    cm2=confusion_matrix(y_test,y_pred2)
    print ('With',i/10,'threshold the Confusion Matrix is ','\n',cm2,'\n',
            'with',cm2[0,0]+cm2[1,1],'correct predictions and',cm2[1,0],'Type II errors( False Negatives)','\n\n',
           'Sensitivity: ',cm2[1,1]/(float(cm2[1,1]+cm2[1,0])),'Specificity: ',cm2[0,0]/(float(cm2[0,0]+cm2[0,1])),'\n
\n\n')
```

```
With 0.1 threshold the Confusion Matrix is
 [[311 348]
 [ 12  80]]
 with 391 correct predictions and 12 Type II errors( False Negatives)

 Sensitivity:  0.8695652173913043 Specificity:  0.47192716236722304



With 0.2 threshold the Confusion Matrix is
 [[518 141]
 [ 43  49]]
 with 567 correct predictions and 43 Type II errors( False Negatives)

 Sensitivity:  0.532608695652174 Specificity:  0.7860394537177542



With 0.3 threshold the Confusion Matrix is
 [[600  59]
 [ 64  28]]
 with 628 correct predictions and 64 Type II errors( False Negatives)

 Sensitivity:  0.30434782608695654 Specificity:  0.9104704097116844



With 0.4 threshold the Confusion Matrix is
 [[640  19]
 [ 80  12]]
 with 652 correct predictions and 80 Type II errors( False Negatives)

 Sensitivity:  0.13043478260869565 Specificity:  0.9711684370257967
```

ROC Curve

In [ ]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_yes[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```



Area under the curve (AUC)

The area under the ROC curve quantifies model classification accuracy; the higher the area, the greater the disparity between true and false positives. The stronger the model in classifying members of the training dataset. An area of 0.5 corresponds to a model that performs no better than random classification, and a good classifier stays as far away from that as possible. An area of 1 is ideal—the closer the AUC to 1, the better.

In [ ]:
```
sklearn.metrics.roc_auc_score(y_test,y_pred_prob_yes[:,1])
```

Out[ ]:
```
0.7386191198786038
```

Conclusion:

- All attributes selected after the elimination process show P-values lower than 5% and thereby suggesting a significant role in the Heart disease prediction.
- Men seem to be more susceptible to heart disease than women. An increase in age, the number of cigarettes smoked per day, and systolic Blood Pressure also shows increased odds of having heart disease.
- Total cholesterol shows no significant change in the odds of CHD. This could be due to good cholesterol(HDL) in total cholesterol reading. Glucose too causes a very negligible change in odds (0.2%)
- The model predicted with 0.87 accuracies. The model is more specific than sensitive.
- The Area under the ROC curve is 73.8, which is somewhat satisfactory.
- The overall model could be improved with more data.

Scenario 2: Titanic Survival Prediction Using Logistic Regression

Industry: Insurance

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone on board, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (i.e. name, age, gender, socio-economic class, etc.).

Dataset Description:

- **pclass**: Ticket class [1 = 1st, 2 = 2nd, 3 = 3rd]
- **sex**: Sex
- **Age**: Age in years(Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5)
- **sibsp**: # of siblings / spouses aboard the Titanic (Sibling = brother, sister, stepbrother, stepsister; Spouse = husband, wife (mistresses and fiancés were ignored)
- **parch**: # of parents / children aboard the Titanic (Parent = mother, father; Child = daughter, son, stepdaughter, stepson; Some children travelled only with a nanny, therefore parch=0 for them.
- **ticket**: Ticket number
- **fare**: Passenger fare
- **cabin**: Cabin number
- **embarked**: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

Target Variable:

- **survival**: Survival [0 = No, 1 = Yes]

Tasks to be Performed:

- Import the required libraries- Beginner
- Prepare the data (Data pre-processing)- Beginner
- Check for missing value- Beginner
- Explore the data using EDA- Intermediate
- Training a logistic regression model- Advance (Bridging Question)
- Evaluating the model- Advance (Bridging Question)

Topics Covered:

- Logistic Regression

Question 1: Import the Required Libraries

In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, roc_auc_score
%matplotlib inline
```

Question 2: Preparing the Data

In [ ]:
```
!wget https://www.dropbox.com/s/rrpaaz4mo3gmxwm/titainic_train.csv -nv
```

```
2020-07-20 07:29:06 URL:https://uc34ee7e2779837094f97fff98b3.dl.dropboxusercontent.com/cd/0/inline/A73kPjtryF_Zs4m9kL
TUSWfQXzNXR0aTrhR5Cc0qNgWNEr0pI2kjtjhssHjoxY6j82GsxnkYA0xbpqDaNgmOTwYpe9lnihpcqDt5wuREbnAw_dGD9XsqTYH5o59rd6nCblE/fil
e [61194/61194] -> "titainic_train.csv" [1]
```

In [ ]:
```
!wget https://www.dropbox.com/s/yzaqdn6cs773ga1/titanic_test.csv -nv
```

```
2020-07-20 07:22:45 URL:https://uc4b9266506cea044262aab83f60.dl.dropboxusercontent.com/cd/0/inline/A73eXq13tf-c-QrUJf
SqXjzM9ANnHWyJJFGkqZ5I7R0dw-OHp3Di_5P4ZTviQGT-jn2bnv6mljrHwlMUZhJ8UNnwrQ2swJkya2eSM-xEqVd1TlLz4eujlsGH-VwDJy_SkMk/fil
e [28629/28629] -> "titanic_test.csv" [1]
```

In [ ]:
```
train_data=pd.read_csv('titainic_train.csv')
train_data.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [ ]:
```
test_data=pd.read_csv('titanic_test.csv')
test_data.head()
```

Out[ ]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

General Data Information

Information like number of non-null values present, datatype of column and memory usage.

In [ ]:
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [ ]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

Question 3: Check for missing value

Number of missing values in train data

In [ ]:
```
miss_train=pd.DataFrame({'Col_name':train_data.columns,'Missing value?':
                         [any(train_data[x].isnull()) for x in train_data.columns],
                'Count_':[sum(train_data[y].isnull()) for y in train_data.columns],
                         'Percentage':[sum(train_data[y].isnull())/train_data.shape[0] for y in train_data.columns]})
```

In [ ]:
```
miss_train.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
10	Cabin	True	687	0.771044
5	Age	True	177	0.198653
11	Embarked	True	2	0.002245
0	PassengerId	False	0	0.000000
1	Survived	False	0	0.000000
2	Pclass	False	0	0.000000
3	Name	False	0	0.000000
4	Sex	False	0	0.000000
6	SibSp	False	0	0.000000
7	Parch	False	0	0.000000
8	Ticket	False	0	0.000000
9	Fare	False	0	0.000000

In [ ]:
```
c=0
for i in range(len(train_data.index)) :
      if any(train_data.iloc[i].isnull()):
            c+=1

print('Total missing values: %s'%miss_train.Count_.sum())
print('Number of rows with missing values: %s'%c)
print('Percentage of records with missing value: ',round(float(c)/int(train_data.shape[0]),2))
```

```
Total missing values: 866
Number of rows with missing values: 708
Percentage of records with missing value:  0.79
```

Number of missing values in test data

In [ ]:
```
miss_test=pd.DataFrame({'Col_name':test_data.columns,'Missing value?':
                        [any(test_data[x].isnull()) for x in test_data.columns],
                 'Count_':[sum(test_data[y].isnull()) for y in test_data.columns],
                       'Percentage':[sum(test_data[y].isnull())/test_data.shape[0] for y in test_data.columns]})
miss_test.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
9	Cabin	True	327	0.782297
4	Age	True	86	0.205742
8	Fare	True	1	0.002392
0	PassengerId	False	0	0.000000
1	Pclass	False	0	0.000000
2	Name	False	0	0.000000
3	Sex	False	0	0.000000
5	SibSp	False	0	0.000000
6	Parch	False	0	0.000000
7	Ticket	False	0	0.000000
10	Embarked	False	0	0.000000

In [ ]:
```
c=0
for i in range(len(test_data.index)) :
      if any(test_data.iloc[i].isnull()):
            c+=1

print('Total missing values: %s'%miss_test.Count_.sum())
print('Number of rows with missing values: %s'%c)
print('Percentage of records with missing value: ',round(float(c)/int(test_data.shape[0]),2))
```

```
Total missing values: 414
Number of rows with missing values: 331
Percentage of records with missing value:  0.79
```

As more than 70% data is missing due to column *cabin* we can drop this column. Then drop the rows with missing data.

In [ ]:
```
train_data.drop('Cabin',axis=1,inplace=True)
```

In [ ]:
```
test_data.drop('Cabin',axis=1,inplace=True)
```

In [ ]:
```
train_data.dropna(inplace=True)
test_data.dropna(inplace=True)
```

In [ ]:
```
train_data.shape,test_data.shape
```

Out[ ]:
```
((712, 11), (331, 10))
```

Testing for missing data again

In [ ]:
```
miss_train=pd.DataFrame({'Col_name':train_data.columns,'Missing value?':
                         [any(train_data[x].isnull()) for x in train_data.columns],
                 'Count_':[sum(train_data[y].isnull()) for y in train_data.columns],
                         'Percentage':[sum(train_data[y].isnull())/train_data.shape[0] for y in train_data.columns]})
miss_train.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	PassengerId	False	0	0.0
1	Survived	False	0	0.0
2	Pclass	False	0	0.0
3	Name	False	0	0.0
4	Sex	False	0	0.0
5	Age	False	0	0.0
6	SibSp	False	0	0.0
7	Parch	False	0	0.0
8	Ticket	False	0	0.0
9	Fare	False	0	0.0
10	Embarked	False	0	0.0

In [ ]:
```
miss_test=pd.DataFrame({'Col_name':test_data.columns,'Missing value?':
                        [any(test_data[x].isnull()) for x in test_data.columns],
                    'Count_':[sum(test_data[y].isnull()) for y in test_data.columns],
                         'Percentage':[sum(test_data[y].isnull())/test_data.shape[0] for y in test_data.columns]})
miss_test.sort_values(by='Count_',ascending=False)
```

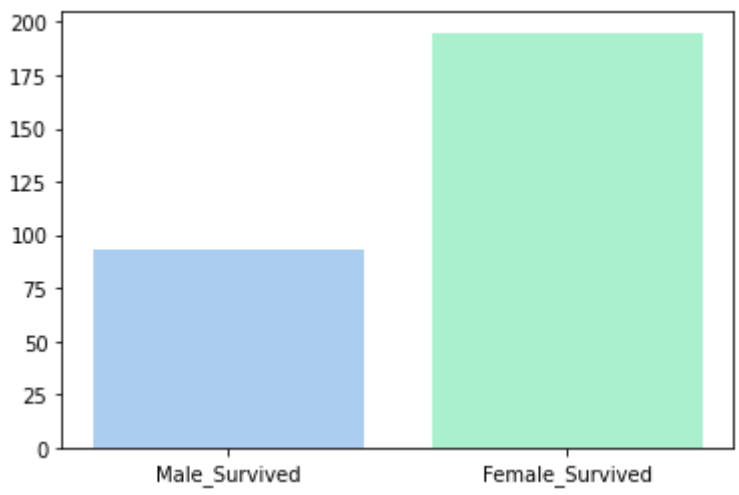
Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	PassengerId	False	0	0.0
1	Pclass	False	0	0.0
2	Name	False	0	0.0
3	Sex	False	0	0.0
4	Age	False	0	0.0
5	SibSp	False	0	0.0
6	Parch	False	0	0.0
7	Ticket	False	0	0.0
8	Fare	False	0	0.0
9	Embarked	False	0	0.0

Question 4: Explore the data using EDA

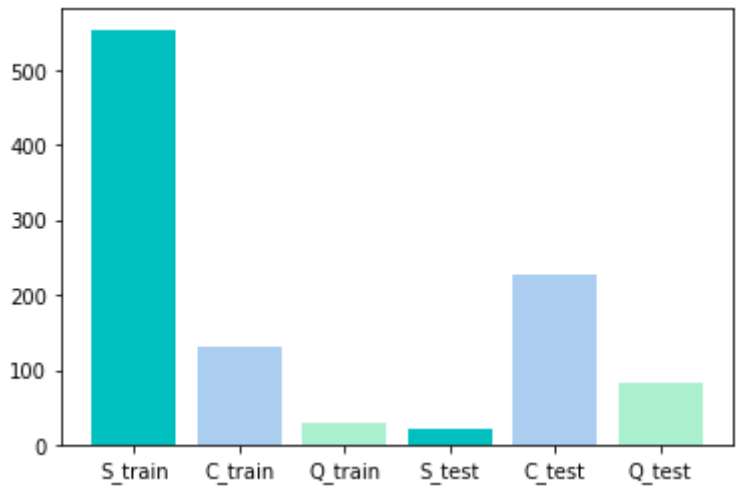
In [ ]:
```
male_survived=len(train_data[(train_data.Sex=='male') & (train_data.Survived==1)])
female_survived=len(train_data[(train_data.Sex=='female') & (train_data.Survived==1)])
```

In [ ]:
```
plt.bar(['Male_Survived','Female_Survived'],[male_survived,female_survived],color=['#abcdef','#abf0ce'])
plt.show()
```



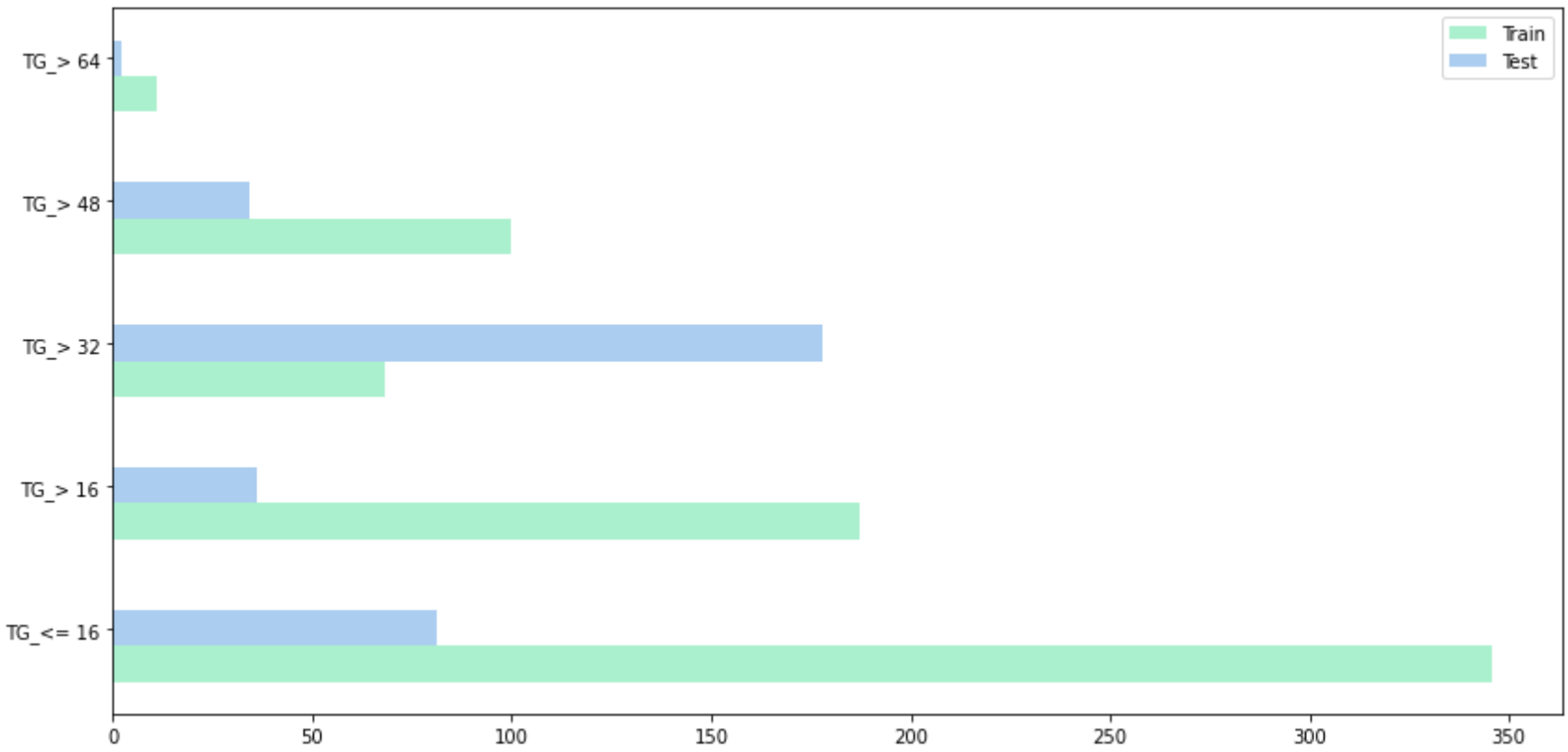
In [ ]:
```
scq=[]
for i in train_data.Embarked.unique():
    scq.append(len(train_data[train_data.Embarked==i]))

for i in test_data.Embarked.unique():
    scq.append(len(test_data[test_data.Embarked==i]))
plt.bar(['S_train','C_train','Q_train','S_test','C_test','Q_test'],scq,color=['c','#abcdef','#abf0ce'])
```

Out[ ]: <BarContainer object of 6 artists>



In [ ]:
```
#  Age Grouping
train_data.loc[ train_data['Age'] <= 16, 'Age'] = 0
train_data.loc[(train_data['Age'] > 16) & (train_data['Age'] <= 32), 'Age'] = 1
train_data.loc[(train_data['Age'] > 32) & (train_data['Age'] <= 48), 'Age'] = 2
train_data.loc[(train_data['Age'] > 48) & (train_data['Age'] <= 64), 'Age'] = 3
train_data.loc[ train_data['Age'] > 64, 'Age'] = 4

test_data.loc[ test_data['Age'] <= 16, 'Age'] = 0
test_data.loc[(test_data['Age'] > 16) & (test_data['Age'] <= 32), 'Age'] = 1
test_data.loc[(test_data['Age'] > 32) & (test_data['Age'] <= 48), 'Age'] = 2
test_data.loc[(test_data['Age'] > 48) & (test_data['Age'] <= 64), 'Age'] = 3
test_data.loc[ test_data['Age'] > 64, 'Age'] = 4
```

In [ ]:
```
agg_train,agg_test=[],[]
for i in train_data.Age.unique():
    agg_train.append(len(train_data[train_data.Age==i]))

for i in test_data.Age.unique():
    agg_test.append(len(test_data[test_data.Age==i]))

age_grps=['TG_<= 16','TG_> 16','TG_> 32','TG_> 48','TG_> 64']
plt.figure(figsize=(14,7))
plt.barh(np.arange(5),agg_train,color='#abf0ce',height=0.25)
plt.barh(np.arange(5)+0.25,agg_test,color='#abcdef',height=0.25)
plt.yticks(np.arange(5) +  0.25,labels=age_grps)
plt.legend(['Train','Test'])
plt.show()
```



In [ ]:
```
# Title
import re
def get_title(name):
    title_search = re.search(' ([A-Za-z]+\.)', name)

    if title_search:
        return title_search.group(1)
    return ""
```

In [ ]:
```
train_data['Title'] = train_data['Name'].apply(get_title)
test_data['Title'] = test_data['Name'].apply(get_title)
```

In [ ]:
```
train_data['Title'].value_counts()
```

Out[ ]:
```
Mr.          398
Miss.        145
Mrs.         107
Master.       36
Rev.           6
Dr.            6
Col.           2
Major.         2
Mlle.          2
Countess.      1
Don.           1
Sir.           1
Lady.          1
Capt.          1
Mme.           1
Ms.            1
Jonkheer.      1
Name: Title, dtype: int64
```

In [ ]:
```
test_data.Title.value_counts()
```

Out[ ]:
```
Mr.        182
Miss.       64
Mrs.        62
Master.     17
Rev.         2
Col.         2
Dr.          1
Dona.        1
Name: Title, dtype: int64
```

In [ ]:
```
train_data['Title'] = train_data['Title'].replace(['Capt.', 'Dr.', 'Major.', 'Rev.'], 'Officer.')
train_data['Title'] = train_data['Title'].replace(['Lady.', 'Countess.', 'Don.', 'Sir.', 'Jonkheer.', 'Dona.'], 'Roya
l.')
train_data['Title'] = train_data['Title'].replace(['Mlle.', 'Ms.'], 'Miss.')
train_data['Title'] = train_data['Title'].replace(['Mme.'], 'Mrs.')
train_data['Title'].value_counts()
```

Out[ ]:
```
Mr.         398
Miss.       148
Mrs.        108
Master.      36
Officer.     15
Royal.        5
Col.          2
Name: Title, dtype: int64
```

In [ ]:
```
test_data['Title'] = test_data['Title'].replace(['Capt.', 'Dr.', 'Major.', 'Rev.'], 'Officer.')
test_data['Title'] = test_data['Title'].replace(['Lady.', 'Countess.', 'Don.', 'Sir.', 'Jonkheer.', 'Dona.'], 'Royal.'
)
test_data['Title'] = test_data['Title'].replace(['Mlle.', 'Ms.'], 'Miss.')
test_data['Title'] = test_data['Title'].replace(['Mme.'], 'Mrs.')
test_data['Title'].value_counts()
```

Out[ ]:
```
Mr.         182
Miss.        64
Mrs.         62
Master.      17
Officer.      3
Col.          2
Royal.        1
Name: Title, dtype: int64
```

In [ ]:
```
#Drop unwanted variables
train_data.drop(['Name', 'Ticket'], axis=1,inplace=True)
test_data.drop(['Name','Ticket'],axis=1,inplace=True)
```

In [ ]:
```
#Create dummies
train_d = pd.get_dummies(train_data, drop_first=True)
test_d=pd.get_dummies(test_data, drop_first=True)
```

Question 5: Train a Logistic Model

In [ ]:
```
from sklearn.linear_model import LogisticRegression
X_train=train_d.drop('Survived',axis=1)
y_train=train_d.Survived
logmodel = LogisticRegression(solver = 'liblinear')
logmodel.fit(X_train,y_train)
```

Out[ ]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [ ]:
```
y_pred=logmodel.predict(test_d)
```

Question 6: Model Evaluation

Model Accuracy

In [ ]:
```
!wget https://www.dropbox.com/s/56r45fh8de58yqg/True_Survived.csv -nv
```

```
2020-07-20 07:29:36 URL:https://uce9d04435cf8a846bb318649588.dl.dropboxusercontent.com/cd/0/inline/A71-kEmQl_j1fzdVVU
mVCLbk83KrpJRX4QHjFpmW4Y50uAuVCg510HRR6lQz-cL7rqWip5Zdv-jwDzSZ9Asbxi14NJOaBMAYBrEd5Nev-DNklbdSS2FjP84I6DladpD2jTQ/fil
e [671/671] -> "True_Survived.csv" [1]
```

In [ ]:
```
y_test=pd.read_csv('True_Survived.csv')
```

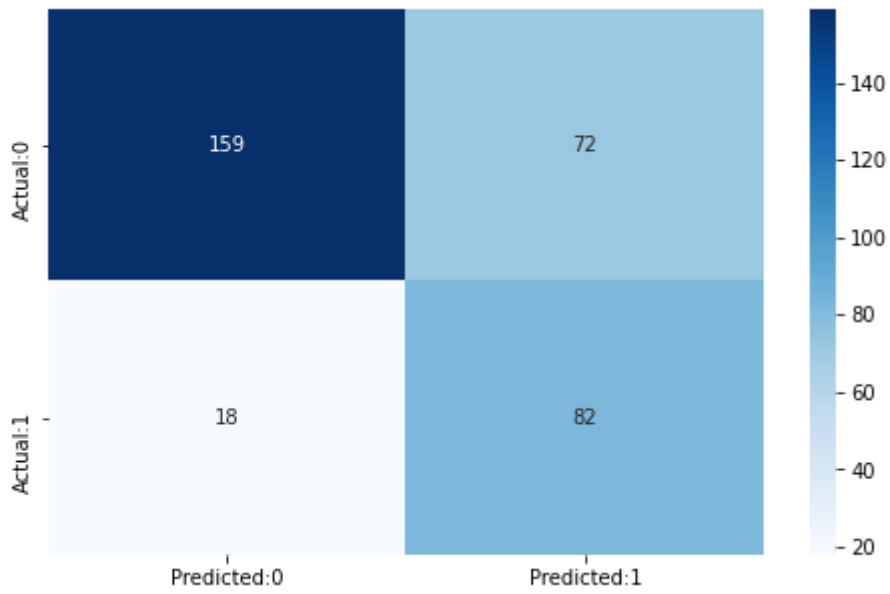
In [ ]:
```
print('The accuracy of the model is: ',round(accuracy_score(y_test,y_pred)*100,2))
```

```
The accuracy of the model is:  72.81
```

Confusion Matrix

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap='Blues')
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8055396a0>
```



In [ ]:
```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 159
True Positive 82
False Negative 18
False Positive 72
Sensitivity 0.82
Specificity 0.6883116883116883
```

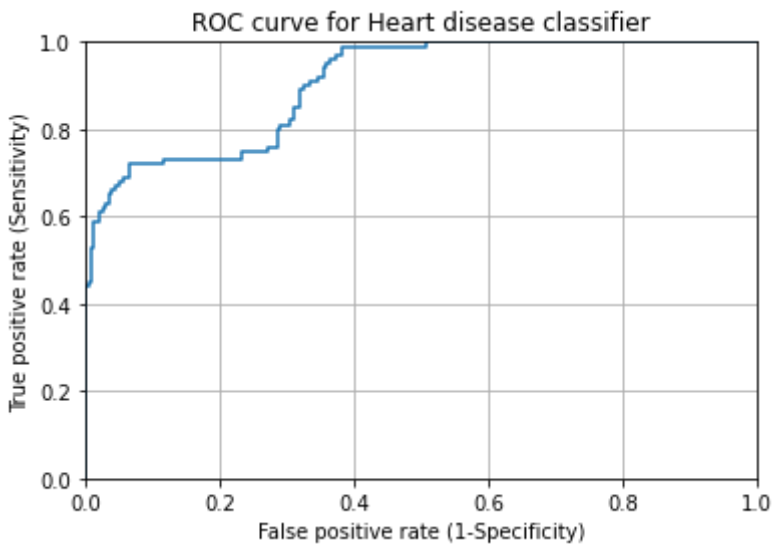
In [ ]:
```
print('The accuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n\n',

'Misclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)
```

```
The accuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.7280966767371602

 Misclassification = 1-Accuracy =  0.27190332326283984

 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.82

 Specificity or True Negative Rate = TN/(TN+FP) =  0.6883116883116883

 Positive Predictive value = TP/(TP+FP) =  0.5324675324675324

 Negative predictive Value = TN/(TN+FN) =  0.8983050847457628

 Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  2.6308333333333334

 Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.2615094339622642
```

The model is more sensitive than specific means it predicts survival more accurately.

ROC Curve

In [ ]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, logmodel.predict_proba(test_d)[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```



Area under the curve (AUC)

In [ ]:
```
print('The Area under the curve is: ',round(roc_auc_score(y_test,logmodel.predict_proba(test_d)[:,1])*100,2))
```

```
The Area under the curve is:  90.41
```

As discussed earlier, if the AUC value is near to 1 it is better as around 90% of data is correctly predicted by the model.

Conclusion:

- The number of females survived is more as compared to males survived.
- The model predicted with 73.11 accuracies. The model is more specific than sensitive.
- The Area under the ROC curve is 90.42, which is somewhat satisfactory.
- The overall model could be improved with more data.

Scenario 3: EduFun Nursery

Industry: Educational

EduFun nursery is one of the top nurseries in Philadelphia. Every year thousands of parents apply for their child's admission here. Due to the excess applications, there was an objective to reject these applications with proper response too. So the board decided to classify the applications based on the occupation of parents and child's nursery, family structure and financial standing, and social and health picture of the family.

Tasks to be Performed:

- Import the required libraries- Beginner
- Prepare the data (Data Preprocessing)- Intermediate
- Check for missing value- Beginner
- Explore the data using EDA- Intermediate
- Training a logistic regression model- Advance (Bridging Question)
- Evaluating the model- Intermediate (Bridging Question)

Variable Attributes

- Employment of parents and child's nursery
  - **parents**: Parents' occupation[usual, pretentious, great_pret]
  - **has_nurs**: Child's nursery[proper, less_proper, improper, critical, very_crit]
- Family structure and financial standings
  - Family structure
    - **form**: Form of the family[complete, completed, incomplete, foster]
    - **children**: Number of children[1, 2, 3, more]
    - **housing**: Housing conditions[convenient, less_conv, critical]
    - **finance**: Financial standing of the family[convenient, inconv]
- Social and health picture of the family
  - **social**: Social conditions[non-prob, slightly_prob, problematic]
  - **health**: Health conditions[recommended, priority, not_recom]
- Target Variable
  - **NURSERY**: Evaluation of applications for nursery schools

```
 class        N         N[%]
------------------------------
not_recom    4320   (33.333 %)
recommend       2   ( 0.015 %)
very_recom    328   ( 2.531 %)
priority     4266   (32.917 %)
spec_prior   4044   (31.204 %)
```

Question 1: Import the Required Libraries

In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, roc_auc_score
%matplotlib inline
```

Question 2: Preparing the Data

In [ ]:
```
!wget https://www.dropbox.com/s/k2s14n5dp0phcyg/nursery.csv -nv
```

```
2020-07-20 07:29:46 URL:https://ucc17c2b7d72e8198b43bcf0321e.dl.dropboxusercontent.com/cd/0/inline/A71BN_OKglEC1ULELs
crbnDfnvcvK5pQYHHC9y4C3w5a3ZWp_MQFOd6M1j63ZEh6z96ZoEU0fn47N0SwtcxnLRTkiSmLssPBmo0sE2VSNALenOfKEpJQCl1dQFMxcdpoIb8/fil
e [1059371/1059371] -> "nursery.csv" [1]
```

In [ ]:
```
data=pd.read_csv('nursery.csv',header=None)
```

In [ ]:
```
cols=['parents','has_nurs','form','children','housing','finance','social','health','NURSERY']
data.columns=cols
```

In [ ]:
```
data.head()
```

Out[ ]:

	parents	has_nurs	form	children	housing	finance	social	health	NURSERY
0	usual	proper	complete	1	convenient	convenient	nonprob	recommended	recommend
1	usual	proper	complete	1	convenient	convenient	nonprob	priority	priority
2	usual	proper	complete	1	convenient	convenient	nonprob	not_recom	not_recom
3	usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recommend
4	usual	proper	complete	1	convenient	convenient	slightly_prob	priority	priority

General Data Information

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12960 entries, 0 to 12959
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   parents   12960 non-null  object
 1   has_nurs  12960 non-null  object
 2   form      12960 non-null  object
 3   children  12960 non-null  object
 4   housing   12960 non-null  object
 5   finance   12960 non-null  object
 6   social    12960 non-null  object
 7   health    12960 non-null  object
 8   NURSERY   12960 non-null  object
dtypes: object(9)
memory usage: 911.4+ KB
```

Question 3: Check for Missing Values

In [ ]:
```
miss_train=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                         [any(data[x].isnull()) for x in data.columns],
                'Count_':[sum(data[y].isnull()) for y in data.columns],
                         'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss_train.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	parents	False	0	0.0
1	has_nurs	False	0	0.0
2	form	False	0	0.0
3	children	False	0	0.0
4	housing	False	0	0.0
5	finance	False	0	0.0
6	social	False	0	0.0
7	health	False	0	0.0
8	NURSERY	False	0	0.0

Dataset has no missing values

Question 4: Explore the data using EDA

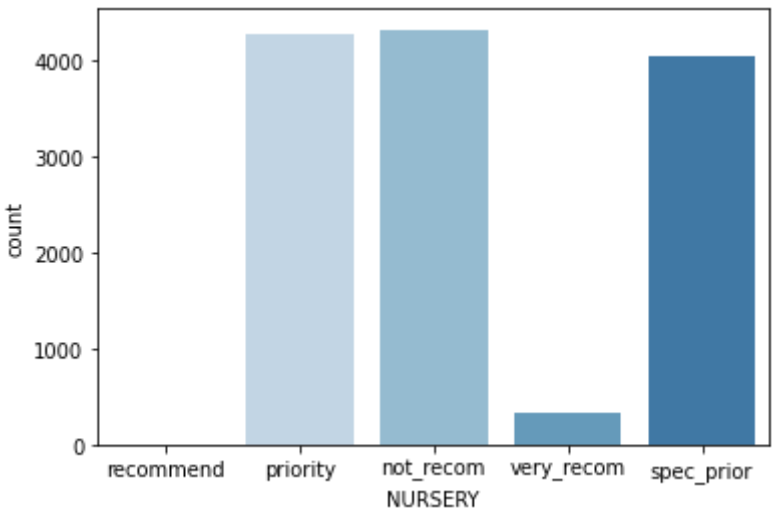
In [ ]:
```
data.head()
```

Out[ ]:

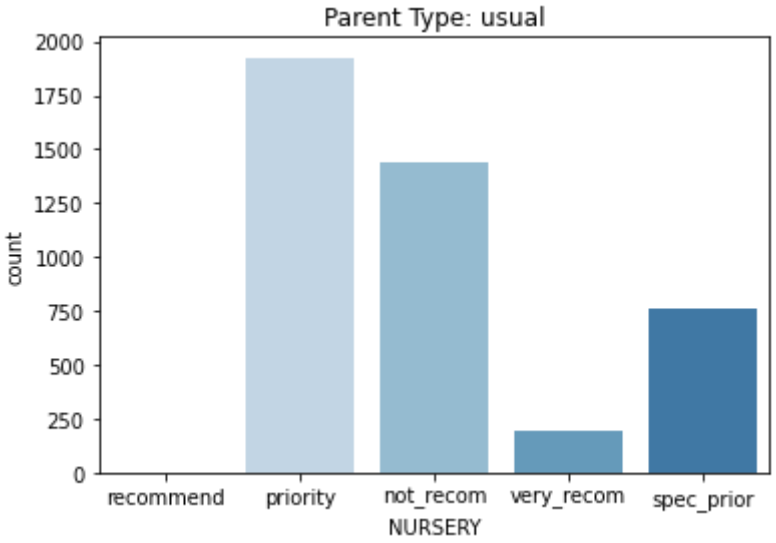
	parents	has_nurs	form	children	housing	finance	social	health	NURSERY
0	usual	proper	complete	1	convenient	convenient	nonprob	recommended	recommend
1	usual	proper	complete	1	convenient	convenient	nonprob	priority	priority
2	usual	proper	complete	1	convenient	convenient	nonprob	not_recom	not_recom
3	usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recommend
4	usual	proper	complete	1	convenient	convenient	slightly_prob	priority	priority

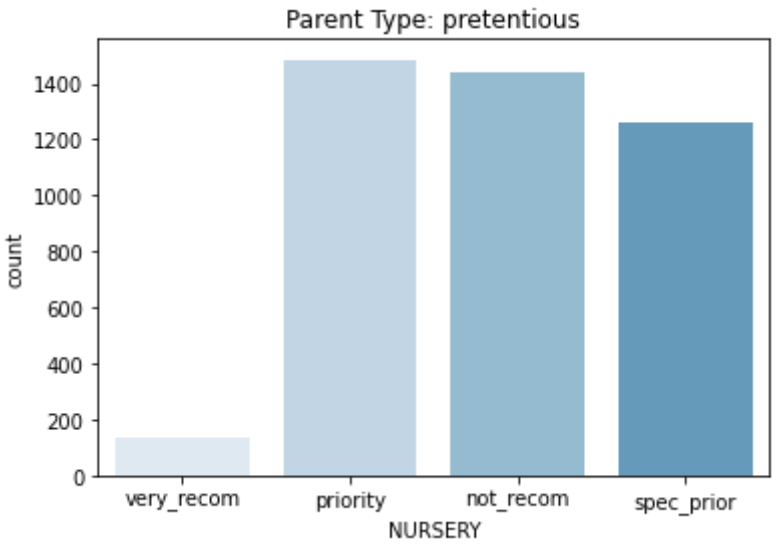
In [ ]:
```
#Nursery application types distibution
sns.set_palette(sns.color_palette("Blues"))
sns.countplot(data.NURSERY,saturation=0.7)
```

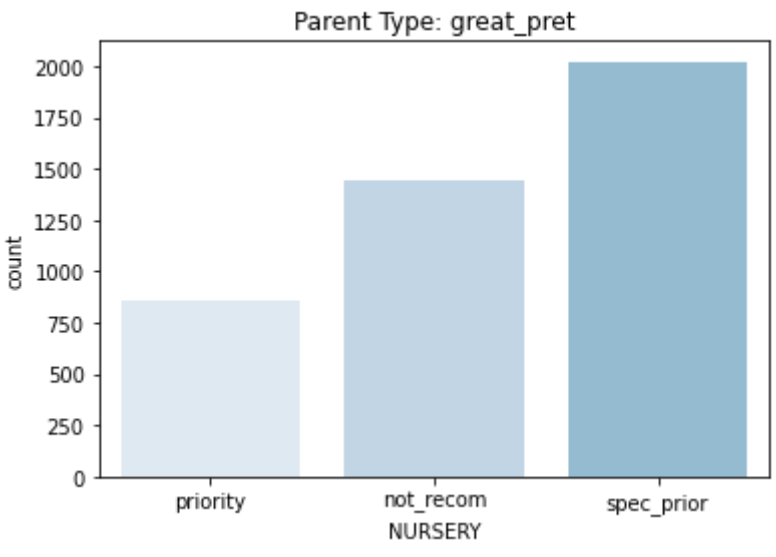
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff80532fb70>



In [ ]:
```
for i in data.parents.unique():
    temp=data[data.parents==i]
    sns.countplot(temp.NURSERY,saturation=0.7)
    plt.title('Parent Type: %s'%i)
    plt.show()
```

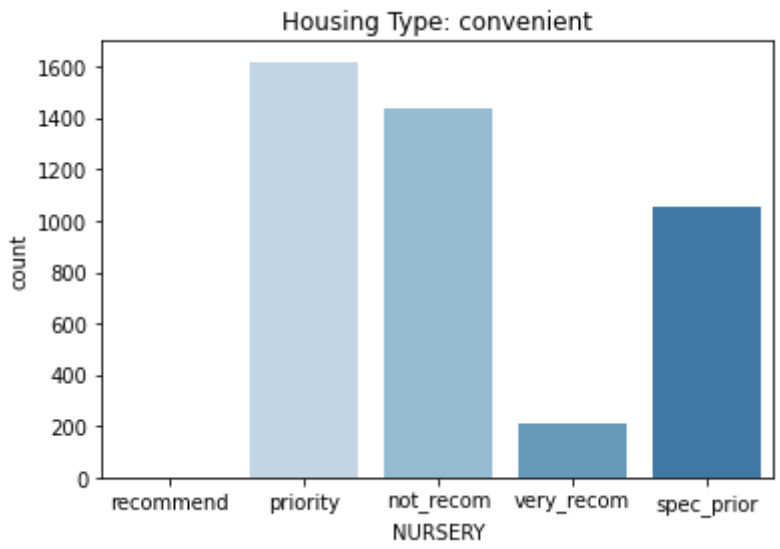


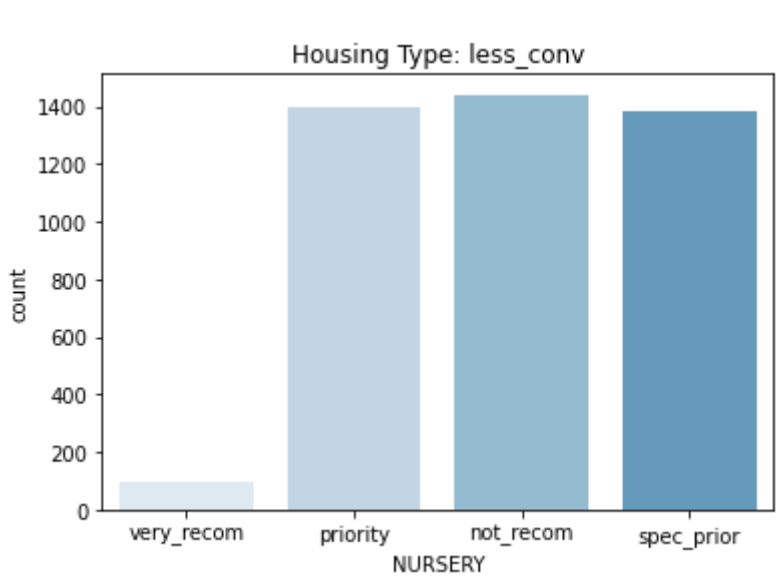


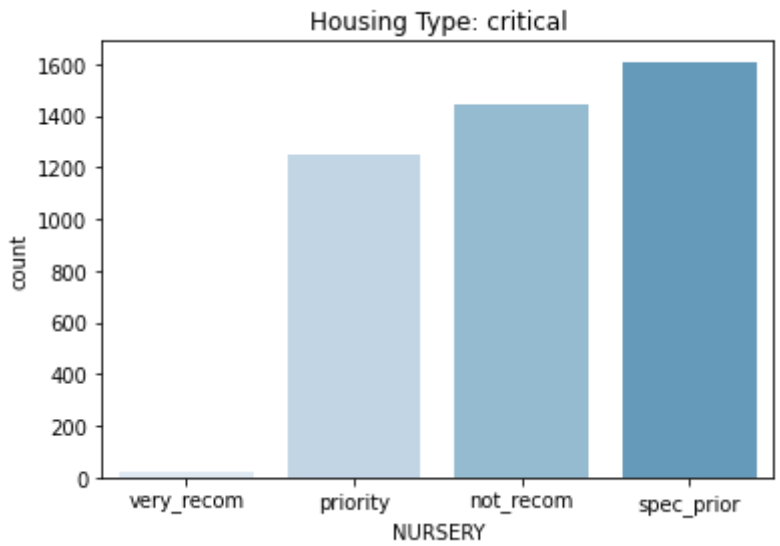


Usual parents get less priority

In [ ]:
```
for i in data.housing.unique():
    temp=data[data.housing==i]
    sns.countplot(temp.NURSERY,saturation=0.7)
    plt.title('Housing Type: %s'%i)
    plt.show()
```

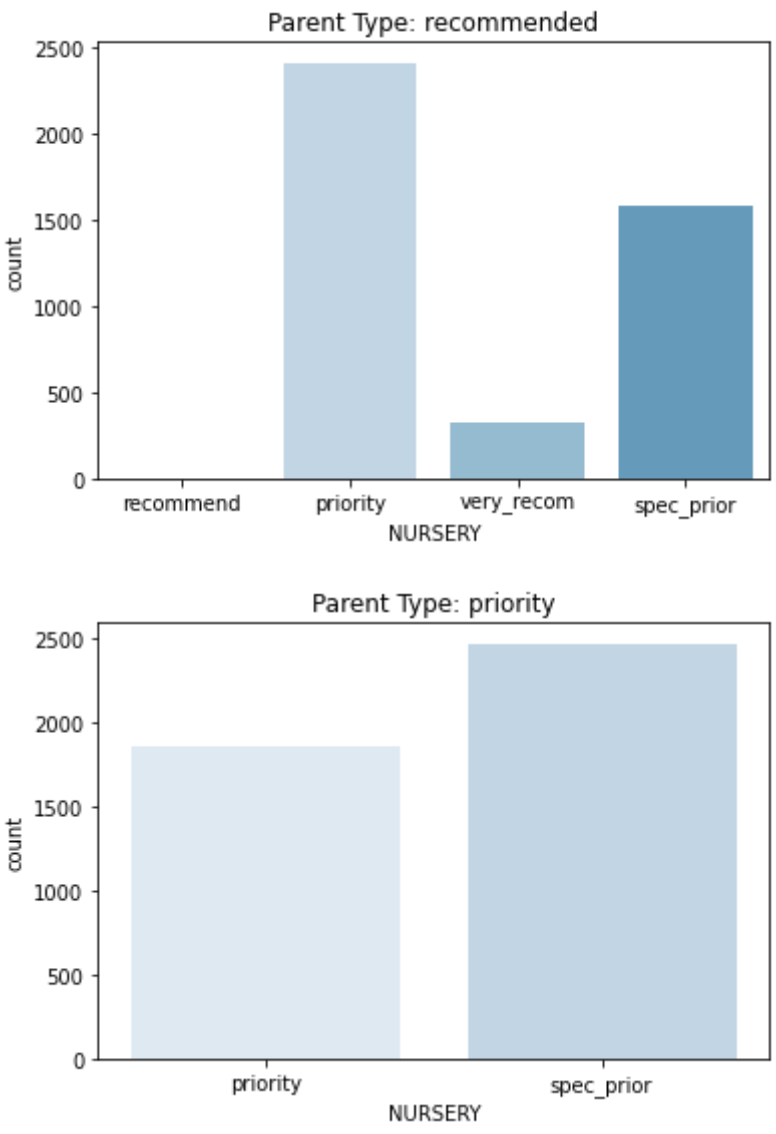


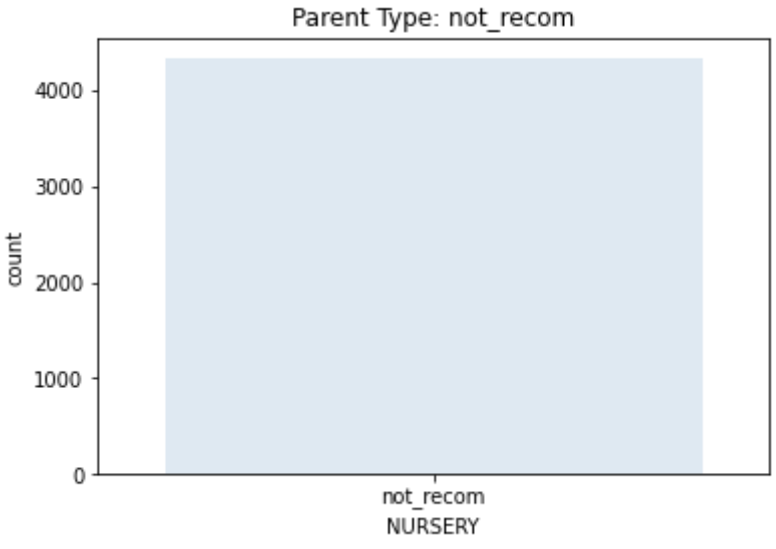




Priority is higher if housing type is critical.

In [ ]:
```
# Dependence on health
for i in data.health.unique():
    temp=data[data.health==i]
    sns.countplot(temp.NURSERY,saturation=0.7)
    plt.title('Parent Type: %s'%i)
    plt.show()
```





We can clearly see that if health is not good then the child is less likely to be accepted.

Question 5: Train a Logistic Model

In [ ]:
```
# Dummies
data_dum=pd.get_dummies(data,columns=cols[:-1])
```

In [ ]:
```
data_dum.head()
```

Out[ ]:

	NURSERY	parents_great_pret	parents_pretentious	parents_usual	has_nurs_critical	has_nurs_improper	has_nurs_less_proper	has_nurs_prope
0	recommend	0	0	1	0	0	0	
1	priority	0	0	1	0	0	0	
2	not_recom	0	0	1	0	0	0	
3	recommend	0	0	1	0	0	0	
4	priority	0	0	1	0	0	0	

In [ ]:
```
# Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test=train_test_split(data_dum.drop('NURSERY',axis=1),data_dum.NURSERY, test_size=0.3,rando
m_state=101)
```

In [ ]:
```
#label encode target variable
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
y_test=lb.fit_transform(y_test)
y_train=lb.fit_transform(y_train)
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression(C=0.5,solver = 'sag',multi_class='multinomial',max_iter=1000)
logmodel.fit(X_train,y_train)
```

Out[ ]:
```
LogisticRegression(C=0.5, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='sag', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [ ]:
```
y_pred=logmodel.predict(X_test)
```

Step 6: Model Evaluation

Model Accuracy

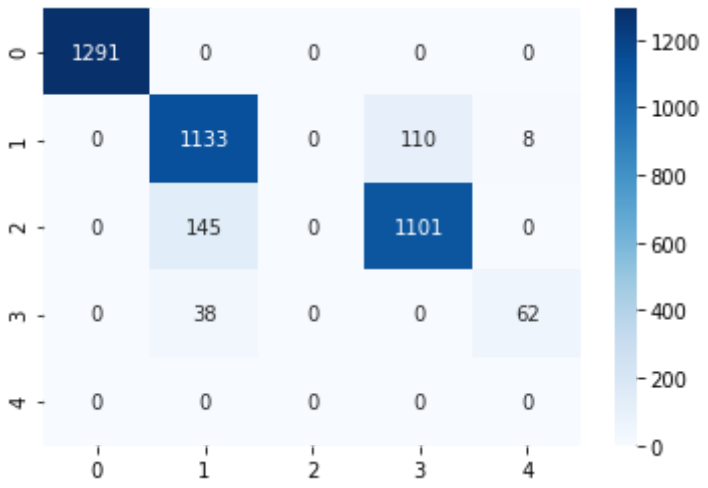
In [ ]:
```
print('The accuracy of the model is: ',round(accuracy_score(y_test,y_pred)*100,2))
```

```
The accuracy of the model is:  62.35
```

Confusion Matrix

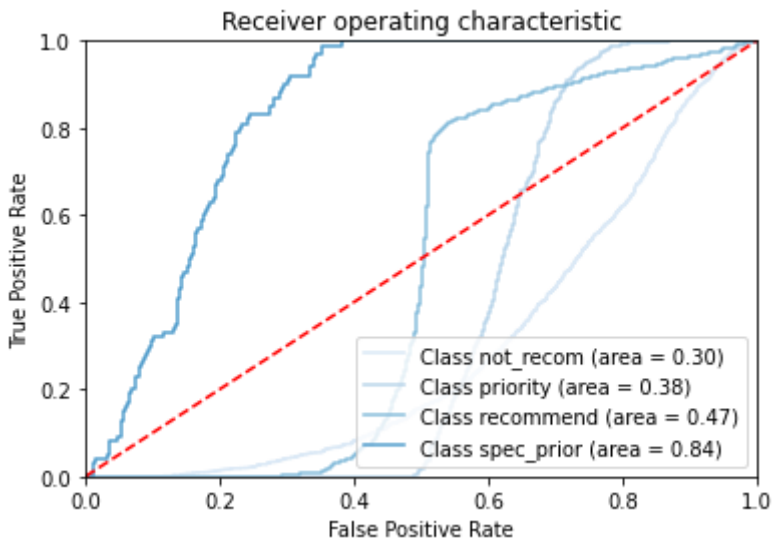
In [ ]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm, annot=True,fmt='d',cmap='Blues')
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff804f0f9b0>
```



ROC Curve

In [ ]:
```
from sklearn.metrics import roc_curve
logmodel_prob=logmodel.predict_proba(X_test).T


dummy_y_test=pd.get_dummies(y_test)

roc_auc=dict()
lfpr4=dict()
ltpr6=dict()
lthresholds4=dict()
for i in dummy_y_test.columns:
    roc_auc[i]=roc_auc_score(dummy_y_test[i],logmodel_prob[i-1])
    lfpr4[i], ltpr6[i], lthresholds4[i] = roc_curve(dummy_y_test[i], logmodel_prob[i-1])


for i in dummy_y_test.columns:
    cls=lb.classes_
    plt.plot(lfpr4[i], ltpr6[i], label='Class '+str(cls[i])+ ' (area = %0.2f)' % roc_auc[i])



plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```



As discussed earlier, if the AUC value is near to 1 it is better as around 90% of data is correctly predicted by the model.

Conclusion:

- Usual parents are not much prioritized.
- Health is directly related to acceptance.
- Housing is inversely related to acceptance.
- The model predicted with 62.35 accuracy. The model is more specific than sensitive.
- The Area under the ROC curve value for spec_prior class is .84, which is better but for other class can be improved.
- The overall model could be improved with more data.

Scenario 4: Online Shopper Intention

Industry: Business

The dataset is collected from 12330 online sessions of multiple users around 1-year long, avoiding redundant or similar courses. Several features were recorded during each session, like administrative, informational, product-related, and respective durations. "Special Day" feature indicates the closeness of the site visiting time to a specific special day (e.g., Mother's Day, Valentine's Day) in which the sessions are more likely to be finalized with the transaction. This attribute's value is determined by considering the dynamics of e-commerce, such as the duration between the order date and delivery date. For example, for Valentine's day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8. The dataset also includes an operating system, browser, region, traffic type, visitor type as returning or new visitor, a Boolean value indicating whether the date of the visit is weekend, and month of the year.

The task is to find the generation of revenue or not.

Tasks to be Performed:

- Import the required libraries- Beginner
- Prepare the data (Data Preprocessing)- Intermediate
- Check for missing value- Beginner
- Explore the data using EDA- Intermediate
- Training a logistic regression model- Advance (Bridging Question)
- Evaluating the model- Advance
- Evaluate the model using L2 regression- Advance (Bridging Question)

Variable Attributes

- **Administrative:** This is the number of pages of this type (administrative) that the user visited.
- **Administrative_Duration:** This is the amount of time spent in this category of pages.
- *Informational:* _ This is the number of pages of this type (informational) that the user visited.
- **Informational_Duration:** This is the amount of time spent in this category of pages.
- **ProductRelated:** This is the number of pages of this type (product-related) that the user visited.
- **ProductRelated_Duration:** This is the amount of time spent in this category of pages.
- **BounceRates:** The percentage of visitors who enter the website through that page and exit without triggering any additional tasks.
- **ExitRates:** The percentage of pageviews on the site that end at that specific page.
- **PageValues:** The average value of the page averaged over the amount of the target page and the completion of an eCommerce transaction.
- **SpecialDay:** This value represents the closeness of the browsing date to particular days or holidays (e.g., Mother's Day or Valentine's day) in which the transaction is more likely to be finalized.
- **Month:** Contains the month the pageview occurred, in string form.
- _*OperatingSystems:* An integer value represents the operating system that the user was on when viewing the page.
- **Browser:** An integer value representing the browser that the user was using to view the page.
- **Region:** An integer value representing which region the user is located in.
- **TrafficType:** An integer value representing what type of traffic the user is categorized into.
- **VisitorType:** A string representing whether a visitor is New Visitor, Returning Visitor, or Other.
- **Weekend:** A boolean representing whether the session is on the weekend.
- **Revenue:** A boolean representing whether or not the user completed the purchase.

Question 1: Import the Required Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import sklearn
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn import metrics
from sklearn.metrics import roc_auc_score
from sklearn.metrics import log_loss
```

Question 2: Prepare the Data

In [ ]:
```
!wget https://www.dropbox.com/s/q0q736smgi6cqtr/online_shoppers_intention.csv -nv
```

```
2020-07-20 07:29:54 URL:https://uc8d59a51ab382fff20fe27034a5.dl.dropboxusercontent.com/cd/0/inline/A723LRV4P4oQzBI3ud
qTsy34CfYXnizFHRFCrkPTPxvqxS10yv6K1QIDD8pNj9HSzbLAfCkxipywxSXsuxS1AWl2VarVd2BzQFXUwXOEoHfgRJ-ZzvFbQyd4kmZ2G99QuxI/fil
e [1072063/1072063] -> "online_shoppers_intention.csv" [1]
```

In [ ]:
```
import pandas as pd
data=pd.read_csv('online_shoppers_intention.csv')
```

In [ ]:
```
data.head()
```

Out[ ]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates
0	0	0.0	0	0.0	1	0.000000	0.20	0.20
1	0	0.0	0	0.0	2	64.000000	0.00	0.10
2	0	0.0	0	0.0	1	0.000000	0.20	0.20
3	0	0.0	0	0.0	2	2.666667	0.05	0.14
4	0	0.0	0	0.0	10	627.500000	0.02	0.05

General Information

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

In [ ]:
```
#All the information likr mean, standard deviation and also for categorical variables like count, frequency.
data.describe(include='all')
```

Out[ ]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	1233
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	2.315166	80.818611	0.503569	34.472398	31.731468	1194.746220	0.022191	
std	3.321784	176.779107	1.270156	140.749294	44.475503	1913.669288	0.048488	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	7.000000	184.137500	0.000000	
50%	1.000000	7.500000	0.000000	0.000000	18.000000	598.936905	0.003112	
75%	4.000000	93.256250	0.000000	0.000000	38.000000	1464.157213	0.016813	
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	

Question 3: Checking for Missing Values

In [ ]:
```
miss_train=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                         [any(data[x].isnull()) for x in data.columns],
                'Count_':[sum(data[y].isnull()) for y in data.columns],
                         'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss_train.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	Administrative	False	0	0.0
1	Administrative_Duration	False	0	0.0
16	Weekend	False	0	0.0
15	VisitorType	False	0	0.0
14	TrafficType	False	0	0.0
13	Region	False	0	0.0
12	Browser	False	0	0.0
11	OperatingSystems	False	0	0.0
10	Month	False	0	0.0
9	SpecialDay	False	0	0.0
8	PageValues	False	0	0.0
7	ExitRates	False	0	0.0
6	BounceRates	False	0	0.0
5	ProductRelated_Duration	False	0	0.0
4	ProductRelated	False	0	0.0
3	Informational_Duration	False	0	0.0
2	Informational	False	0	0.0
17	Revenue	False	0	0.0

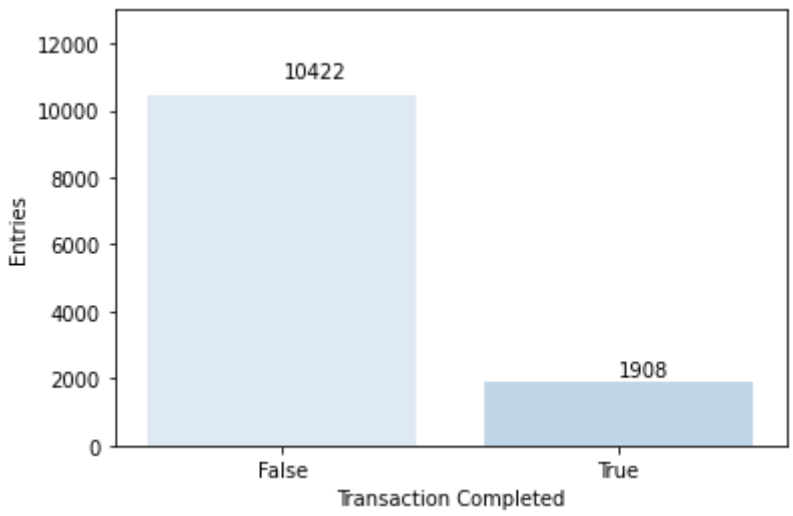
Dataset does not have any missing values.

Question 4: Explore the Data using EDA

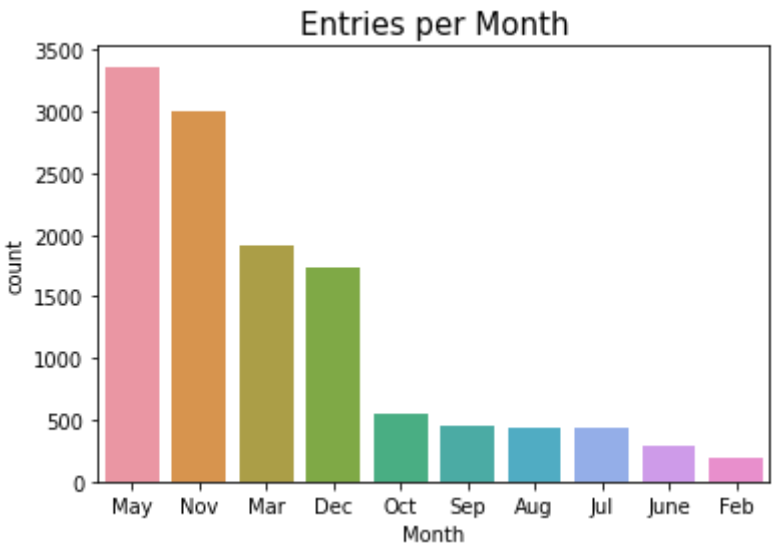
In [ ]:
```
import seaborn as sns
sns.set_palette(sns.color_palette('Blues'))
data.Revenue.value_counts()
```

Out[ ]:
```
False    10422
True      1908
Name: Revenue, dtype: int64
```

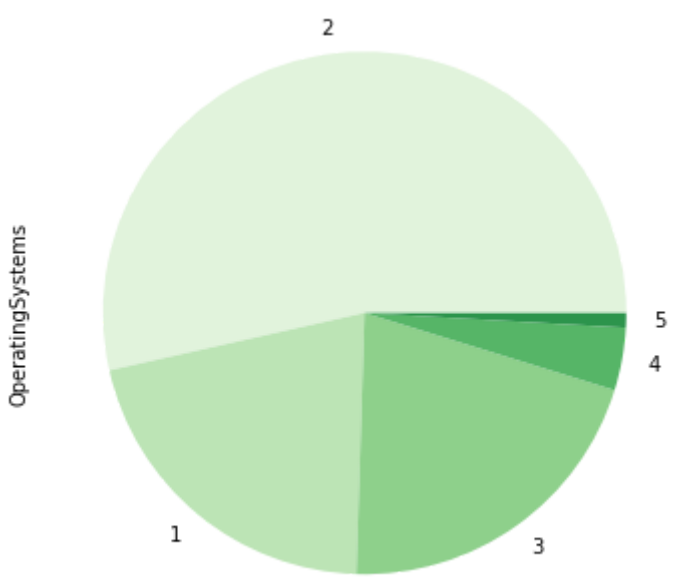
In [ ]:
```
sns.countplot(data['Revenue'])
plt.ylim(0,13000)
plt.text(0, 11000,data.Revenue.value_counts()[0])
plt.text(1,2100,data.Revenue.value_counts()[1])
plt.xlabel('Transaction Completed')
plt.ylabel('Entries')
plt.show()
```



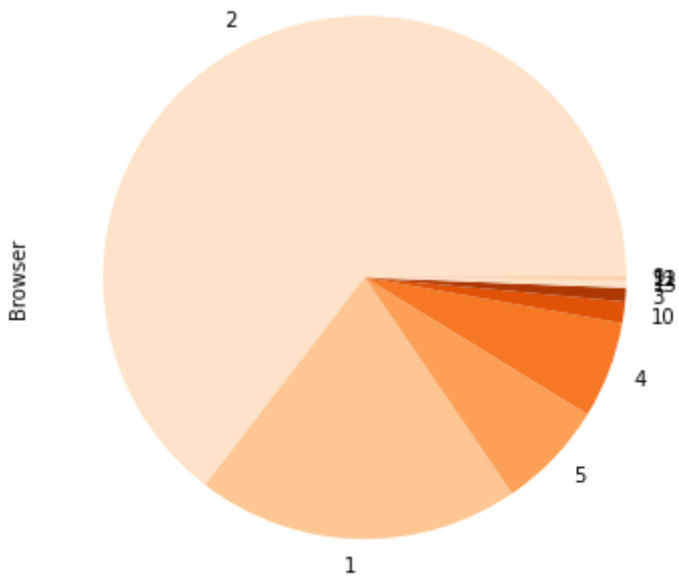
In [ ]:
```
monthly = data['Month'].value_counts()
sns.countplot(data['Month'], order=monthly.index)
plt.title('Entries per Month', fontsize=15)
plt.show()
```



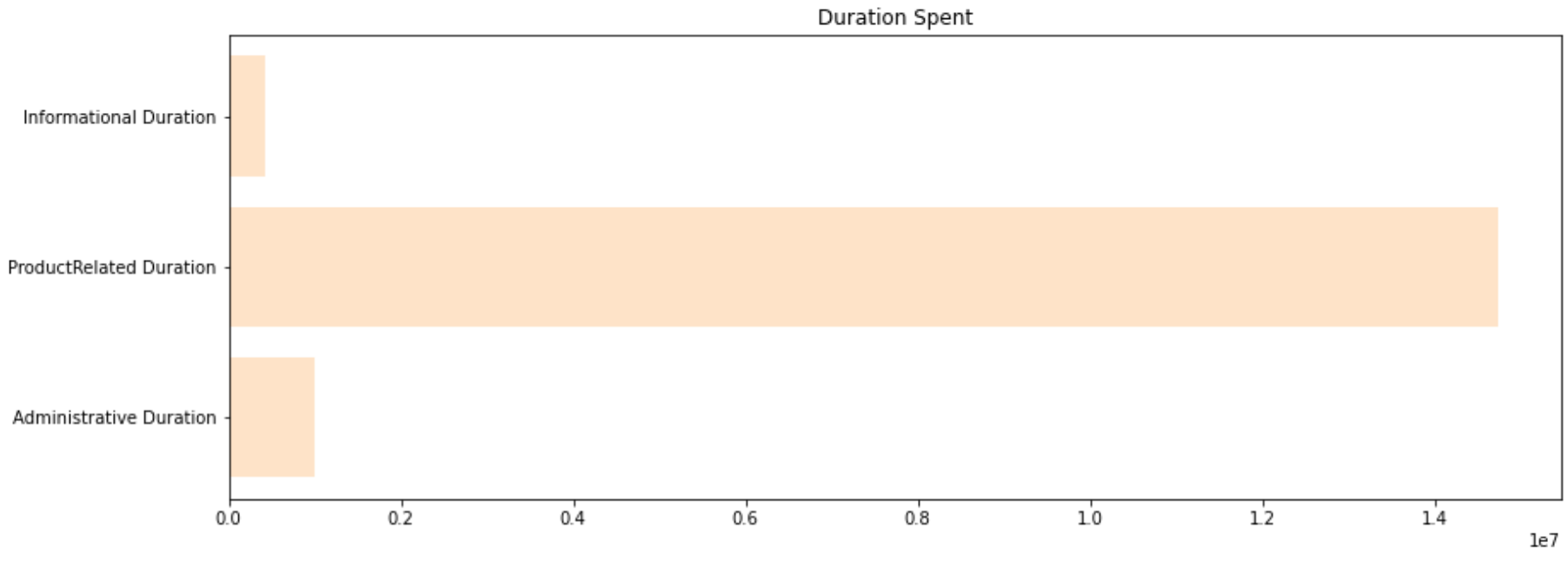
In [ ]:
```
sns.set_palette(sns.color_palette('Greens'))
data['OperatingSystems'] = data['OperatingSystems'].replace([5,6,7,8],5)
os_plot = data['OperatingSystems'].value_counts().plot.pie(figsize=(6,6))
plt.show()
```



In [ ]:
```
sns.set_palette(sns.set_palette('Oranges'))
data['Browser'] = data['Browser'].replace([5,6,7,8],5)
os_plot = data['Browser'].value_counts().plot.pie(figsize=(6,6))
plt.show()
```

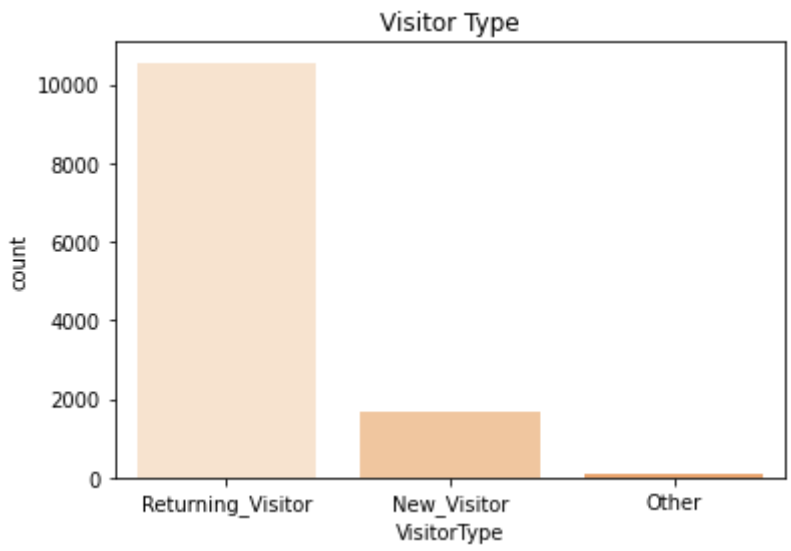


In [ ]:
```
ad=data.Administrative_Duration.sum()
prd=data.ProductRelated_Duration.sum()
ind=data.Informational_Duration.sum()
plt.figure(figsize=(14,5))
plt.barh(['Administrative Duration','ProductRelated Duration', 'Informational Duration'],[ad,prd,ind],)
plt.title('Duration Spent')
plt.show()
```



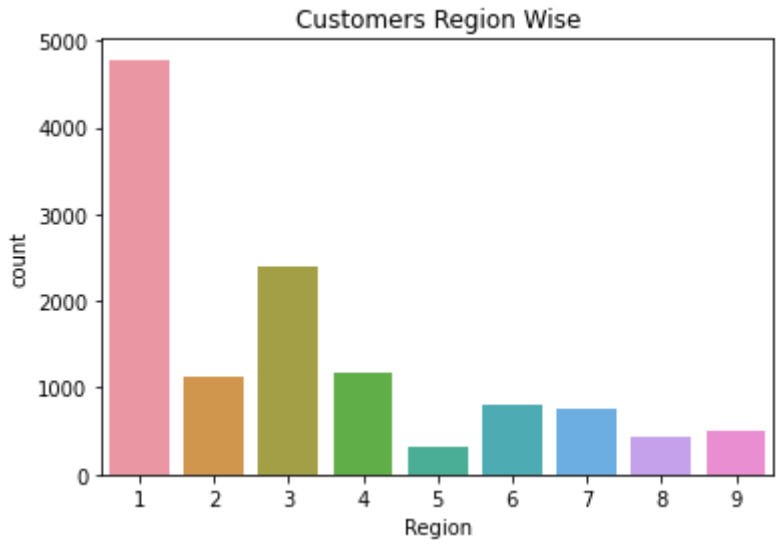
Most of the times is spent on ProductRelated Pages

In [ ]:
```
sns.countplot(data.VisitorType,orient="h")
plt.title('Visitor Type')
plt.show()
```



Most of the visitors are returning type

In [ ]:
```
sns.countplot(data.Region)
plt.title('Customers Region Wise')
plt.show()
```



In [ ]:
```
plt_data={}
for grp,gata in data.groupby('Region'):
    plt_data[grp]=len(gata[gata.Revenue==True])
plt.bar(np.arange(9)+1,list(plt_data.values()))
plt.title('Region Wise Revenue')
plt.show()
```

Question 5: Training a logistic regression model

In [ ]:
```
cols=list(data.columns)
```

In [ ]:
```
data_dum=pd.get_dummies(data,columns=cols[:-1])
data_dum.head()
```

Out[ ]:

	Revenue	Administrative_0	Administrative_1	Administrative_2	Administrative_3	Administrative_4	Administrative_5	Administrative_6	Administra
0	False	1	0	0	0	0	0	0	
1	False	1	0	0	0	0	0	0	
2	False	1	0	0	0	0	0	0	
3	False	1	0	0	0	0	0	0	
4	False	1	0	0	0	0	0	0	

5 rows × 23918 columns

In [ ]:
```
# Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test=train_test_split(data_dum.drop('Revenue',axis=1),data.Revenue, test_size=0.3,random_st
ate=101)

from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression(C=5,solver = 'liblinear',max_iter=100)
logmodel.fit(X_train,y_train)

y_pred=logmodel.predict(X_test)
```

Question 6: Model Evaluation

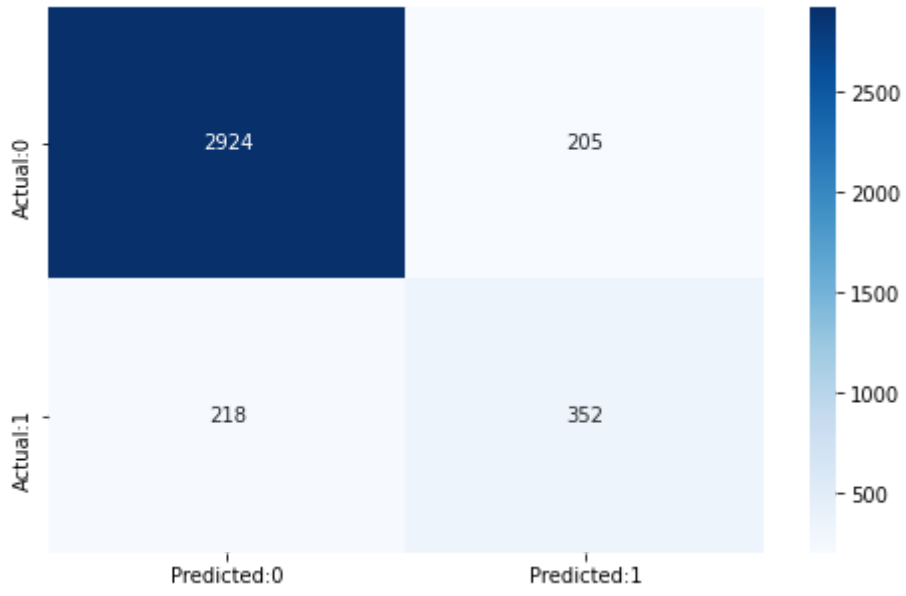
Model Accuracy

In [ ]:
```
print('The accuracy of the model is: ',round(metrics.accuracy_score(y_test,y_pred)*100,2))
```

```
The accuracy of the model is:  88.56
```

Confusion Matrix

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap='Blues')
plt.show()
```



In [ ]:
```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 2924
True Positive 352
False Negative 218
False Positive 205
Sensitivity 0.6175438596491228
Specificity 0.9344838606583573
```

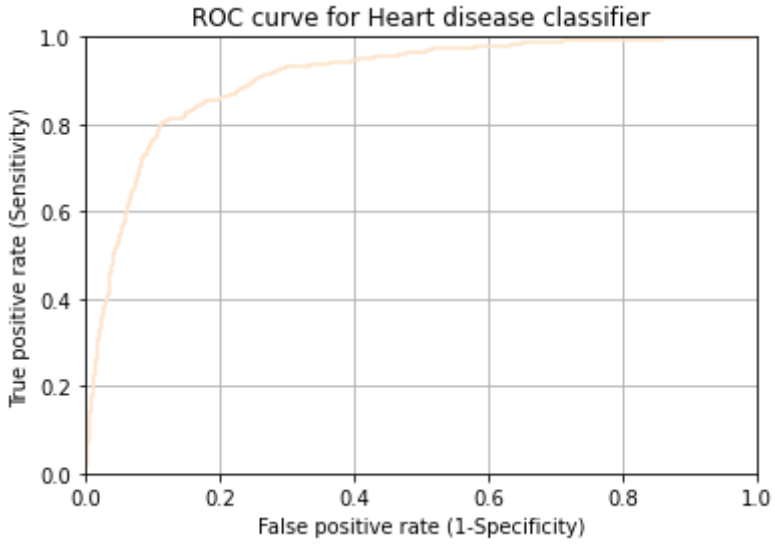
In [ ]:
```
print('The accuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n\n',

'Misclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)
```

```
The accuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.8856447688564477

 Misclassification = 1-Accuracy =  0.11435523114355228

 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.6175438596491228

 Specificity or True Negative Rate = TN/(TN+FP) =  0.9344838606583573

 Positive Predictive value = TP/(TP+FP) =  0.6319569120287253

 Negative predictive Value = TN/(TN+FN) =  0.9306174411203055

 Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  9.425827984595637

 Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.40926992584059324
```

The model is more specific than sensitive means it predicts non-revenue generation accurately.

ROC Curve

In [ ]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, logmodel.predict_proba(X_test)[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
plt.show()
```



Area under the curve (AUC)

In [ ]:
```
print('The Area under the curve is: ',round(roc_auc_score(y_test,logmodel.predict_proba(X_test)[:,1])*100,2))
```

```
The Area under the curve is:  90.52
```

Question-7: Evaluate the model using L2 regression

In [ ]:
```
lr=LogisticRegression(penalty='l2', C=20.0, random_state=911)
lr=lr.fit(X_train, y_train)
predictions = lr.predict(X_train)
accuracy_score(predictions,y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[ ]:
```
0.9998841385702699
```

Conclusion:

- Region 1 produced more revenue as compared to all other regions.
- The model is more sensitive than being specific.
- Returning Visitor is more than other types of visitors, and most of the user's user type 2 operating system and type 2 browser.
- The model accuracy is 88.56, which is pretty good.
- The AUC score is 90.52, which is good.
- The accuracy can be increased by removing correlated features and parameter tuning.

Scenario 5: Telecom Churn Prediction

A telecom company wants you to analyze its data to help retain its customer. You have been provided the 'Telecom Churn' dataset.

Problem Statement:

You have to create a model to predict which customer will switch to other telecom service providers, based on the relevant customer data.

Dataset Description:

The dataset contains 20 features:

State: state of the customer

Account length: how long the account has been active

Area code: Area code

International plan: Does the customer have any international idea or not

Voice mail plan: Does the customer have any voice mail plan or not

Number vmail messages: Number of voice mail messages

Total day minutes: Total day minutes used

Total day calls: Total day calls made

Total day charge: Total day charge

Total eve minutes: Total evening minutes used

Total eve calls: Total evening calls made

Total eve charge: Total evening charge

Total night minutes: Total night minutes used

Total night calls: Total night calls made

Total night charge: Total night charge

Total intl minutes: Total international minutes used

Total intl calls: Total international calls made

Total intl charge: Total international charge

Customer service calls: Number of customer service calls made

Churn: Customer will switch provider or not

Tasks to be performed:

1. Load the data, check its shape and check for null values, check unique values for categorical feature - Beginner
2. Convert categorical to numerical feature, also create a new feature using 'State' column - Intermediate
3. Plot and analyze Correlation - Beginner
4. Split the dataset for training and testing - Beginner
5. Perform K-Fold cross-validation - Intermediate
6. Perform Grid Search cross-validation - Intermediate
7. Train a Random Forest model and perform prediction on test data - Beginner (Bridging Question)
8. Evaluate the model using a classification report and accuracy score - Beginner
9. Find ten best features using trained random forest model and perform K-Fold cross-validation on the new data - Intermediate (Bridging Question)

Topics Covered:

Data collection

Data analysis

Data wrangling/Feature engineering

In [ ]:
```
!wget https://www.dropbox.com/s/55ar5v2hnvy8cx3/datasets-255093-535845-churn-bigml-80.csv
```

```
--2020-07-20 07:30:42--  https://www.dropbox.com/s/55ar5v2hnvy8cx3/datasets-255093-535845-churn-bigml-80.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/55ar5v2hnvy8cx3/datasets-255093-535845-churn-bigml-80.csv [following]
--2020-07-20 07:30:42--  https://www.dropbox.com/s/raw/55ar5v2hnvy8cx3/datasets-255093-535845-churn-bigml-80.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc81fd2276f1ad85e8a8774b25ac.dl.dropboxusercontent.com/cd/0/inline/A736ZQZflHaVMCPiidSIQ_RHu4z67Cwt
Xsr3Xi8f0I57aVCefDyxYUYjHTtrJ6EfIc3HXMbypL5RJrxhhB3PDcDhZ412_ugSE7QU8zR9WWx0xEVVy-sl6mWd8RleF68aQ-M/file# [following]
--2020-07-20 07:30:43--  https://uc81fd2276f1ad85e8a8774b25ac.dl.dropboxusercontent.com/cd/0/inline/A736ZQZflHaVMCPii
dSIQ_RHu4z67CwtXsr3Xi8f0I57aVCefDyxYUYjHTtrJ6EfIc3HXMbypL5RJrxhhB3PDcDhZ412_ugSE7QU8zR9WWx0xEVVy-sl6mWd8RleF68aQ-M/fi
le
Resolving uc81fd2276f1ad85e8a8774b25ac.dl.dropboxusercontent.com (uc81fd2276f1ad85e8a8774b25ac.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to uc81fd2276f1ad85e8a8774b25ac.dl.dropboxusercontent.com (uc81fd2276f1ad85e8a8774b25ac.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 226663 (221K) [text/plain]
Saving to: ‘datasets-255093-535845-churn-bigml-80.csv’

datasets-255093-535 100%[===================>] 221.35K  --.-KB/s    in 0.09s

2020-07-20 07:30:43 (2.51 MB/s) - ‘datasets-255093-535845-churn-bigml-80.csv’ saved [226663/226663]
```

Question-1: Load and analyze the data

Tasks to do:

Load the data in a pandas DataFrame

Have a look at the first five rows

Check if the dataset contains any null values

Check the shape of the dataset

Check the unique values of the categorical columns

In [ ]:
```
import pandas as pd
df=pd.read_csv('/content/datasets-255093-535845-churn-bigml-80.csv')
```

In [ ]:
```
df.head()
```

Out[ ]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3

In [ ]:
```
df.shape
```

Out[ ]:
```
(2666, 20)
```

The dataset contains 2666 rows and 20 features

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2666 entries, 0 to 2665
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   State                   2666 non-null   object 
 1   Account length          2666 non-null   int64  
 2   Area code               2666 non-null   int64  
 3   International plan      2666 non-null   object 
 4   Voice mail plan         2666 non-null   object 
 5   Number vmail messages   2666 non-null   int64  
 6   Total day minutes       2666 non-null   float64
 7   Total day calls         2666 non-null   int64  
 8   Total day charge        2666 non-null   float64
 9   Total eve minutes       2666 non-null   float64
 10  Total eve calls         2666 non-null   int64  
 11  Total eve charge        2666 non-null   float64
 12  Total night minutes     2666 non-null   float64
 13  Total night calls       2666 non-null   int64  
 14  Total night charge      2666 non-null   float64
 15  Total intl minutes      2666 non-null   float64
 16  Total intl calls        2666 non-null   int64  
 17  Total intl charge       2666 non-null   float64
 18  Customer service calls  2666 non-null   int64  
 19  Churn                   2666 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 398.5+ KB
```

We can see none of the columns contain any null values.

3 rows contain dtypes object

The dependent feature 'Churn' contains boolean values

16 columns contain numeric values

In [ ]:
```
print(len(df['State'].unique()))
print(df['International plan'].unique())
print(df['Voice mail plan'].unique())
print(df['Churn'].unique())
```

```
51
['No' 'Yes']
['Yes' 'No']
[False  True]
```

Question-2: We cannot use string objects for prediction, so convert categorical feature to numerical values

Tasks to do:

Convert all the categorical features except State to numerical values

Create new feature 'state_churn' using the feature 'State,' it should contain the ratio of total churn in a given state and total customers of a given state

Create a new feature, 'avg_day_call_dur' using features 'Total day minutes' and 'Total day calls.'

Drop 'Total day minutes' and 'Total day calls'

Similarly create new features 'avg_night_call_dur' and 'avg_eve_call_dur'

In [ ]:
```
df.replace({'International plan':{'No':0,'Yes':1},'Voice mail plan':{'No':0,'Yes':1},'Churn':{False:0,True:1}},inplace=True)
```

In [ ]:
```
df.head()
```

Out[ ]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls
0	KS	128	415	0	1	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3
1	OH	107	415	0	1	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3
2	NJ	137	415	0	0	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5
3	OH	84	408	1	0	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7
4	OK	75	415	1	0	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3

In [ ]:
```
import numpy as np

state_churn = df.groupby(['State'])['Churn'].agg('mean')
state_churn
```

Out[ ]:
```
State
AK    0.069767
AL    0.106061
AR    0.234043
AZ    0.066667
CA    0.208333
CO    0.118644
CT    0.186441
DC    0.111111
DE    0.156863
FL    0.129630
GA    0.163265
HI    0.045455
IA    0.078947
ID    0.089286
IL    0.088889
IN    0.111111
KS    0.192308
KY    0.139535
LA    0.085714
MA    0.153846
MD    0.233333
ME    0.224490
MI    0.224138
MN    0.185714
MO    0.098039
MS    0.229167
MT    0.188679
NC    0.160714
ND    0.090909
NE    0.088889
NH    0.209302
NJ    0.280000
NM    0.090909
NV    0.213115
NY    0.176471
OH    0.151515
OK    0.134615
OR    0.112903
PA    0.222222
RI    0.062500
SC    0.224490
SD    0.122449
TN    0.121951
TX    0.290909
UT    0.133333
VA    0.059701
VT    0.105263
WA    0.208333
WI    0.065574
WV    0.079545
WY    0.121212
Name: Churn, dtype: float64
```

In [ ]:
```
d=dict()
for i in df.State.unique():
  if i not in d:
    d[i]=state_churn[i]
df['state_churn']=df['State'].map(d)
```

In [ ]:
```
df['avg_day_call_dur']=df['Total day minutes']/df['Total day calls']
df.drop(columns=['Total day minutes','Total day calls'],axis=1,inplace=True)
```

In [ ]:
```
df['avg_night_call_dur']=df['Total night minutes']/df['Total night calls']
df.drop(columns=['Total night minutes','Total night calls'],axis=1,inplace=True)
df['avg_eve_call_dur']=df['Total eve minutes']/df['Total eve calls']
df.drop(columns=['Total eve minutes','Total eve calls'],axis=1,inplace=True)
```

In [ ]:
```
df.dropna(inplace=True)
df.isnull().sum()
```

Out[ ]:
```
State                   0
Account length          0
Area code               0
International plan      0
Voice mail plan         0
Number vmail messages   0
Total day charge        0
Total eve charge        0
Total night charge      0
Total intl minutes      0
Total intl calls        0
Total intl charge       0
Customer service calls  0
Churn                   0
state_churn             0
avg_day_call_dur        0
avg_night_call_dur      0
avg_eve_call_dur        0
dtype: int64
```

In [ ]:
```
df.head()
```

Out[ ]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day charge	Total eve charge	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn	state_churn	avg_da
0	KS	128	415	0	1	25	45.07	16.78	11.01	10.0	3	2.70	1	0	0.192308	
1	OH	107	415	0	1	26	27.47	16.62	11.45	13.7	3	3.70	1	0	0.151515	
2	NJ	137	415	0	0	0	41.38	10.30	7.32	12.2	5	3.29	0	0	0.280000	
3	OH	84	408	1	0	0	50.90	5.26	8.86	6.6	7	1.78	2	0	0.151515	
4	OK	75	415	1	0	0	28.34	12.61	8.41	10.1	3	2.73	3	0	0.134615	

Question-3: Plot the correlation and tell which feature will help the most while prediction

Tasks to do:

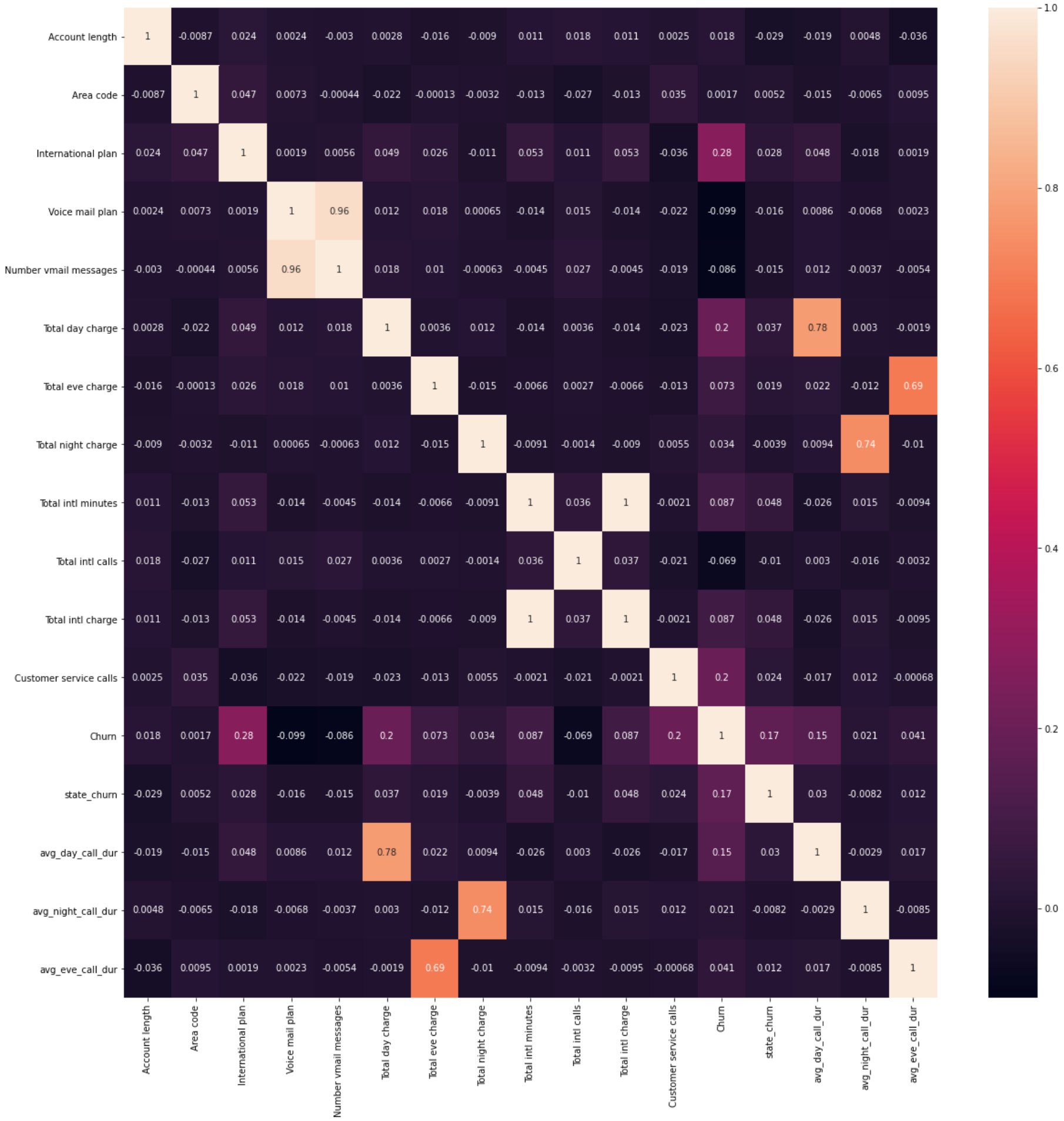
Calculate correlation

Plot the correlation

Compare the correlation

In [ ]:
```
import seaborn as sns
from matplotlib import pyplot as plt
fig, ax = plt.subplots(figsize=(20,20))
correlation = df.corr()
sns.heatmap(correlation,annot=True,ax=ax)
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff8047e69b0>



We can see the new feature 'state_churn' is also highly correlated with the target variable 'churn'

Question-4: Split the data into training and testing datasets

Tasks to do:

Drop the feature 'State' as we cannot use string object for prediction

Split the dataset using sklearn, with 20% for testing with random_state=7

In [ ]:
```
from sklearn.model_selection import train_test_split
X= df.drop(columns=['State','Churn'],axis=1)
y= df['Churn']
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.20,random_state = 7)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2130, 16)
(533, 16)
(2130,)
(533,)
```

Question-5: Perform K-Fold cross-validation

Tasks to do:

Perform Stratified K-fold with K=10

Perform Stratified K-Fold with Random Forest and Decision Tree algorithm

Calculate the mean score of each iteration

- This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score,StratifiedKFold
cv = StratifiedKFold(10)
model_rf = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=7)
model_dt = DecisionTreeClassifier()
score=cross_val_score(model_rf, X_train, y_train, cv=cv)
print(f'score of random forest classifier:{score.mean()}')
score=cross_val_score(model_dt, X_train, y_train, cv=cv)
print(f'score of decision tree classifier:{score.mean()}')
```

```
score of random forest classifier:0.9427230046948356
score of decision tree classifier:0.8971830985915494
```

Since we are getting higher accuracy from random forest, let us train our data random forest model

Question-6: Perform Grid Search cross-validation to find the best parameters for Random Forest Classifier

Tasks to do:

Perform Grid Search CV on random forest classifier to find the best number of:

trees between 20 to 100

maximum depth for the trees, between 3 and 15

maximum features to be used, between range 4 and 17

> Use the following dictionary to implement the first 3 points:
>
> {'n_estimators':range(20,100,20), 'max_depth':range(3,15), 'max_features':range(4,17)}

Perform GridSearchCV with scoring = recall

Cross validate the model with five folds

Check the best score and best estimator found by cross-validation

In [ ]:
```
from sklearn.model_selection import GridSearchCV
model_params = {'n_estimators':range(20,100,20), 'max_depth':range(3,15), 'max_features':range(4,17)}
model_cv = GridSearchCV(model_rf,model_params,cv=5,n_jobs=-1,verbose=True,return_train_score = True,scoring = 'recall'
)
model_cv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 624 candidates, totalling 3120 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:    8.6s
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:   37.0s
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done 1246 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done 1796 tasks      | elapsed:  5.6min
[Parallel(n_jobs=-1)]: Done 2446 tasks      | elapsed:  7.7min
[Parallel(n_jobs=-1)]: Done 3120 out of 3120 | elapsed:  9.8min finished
```

Out[ ]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=-1,
                                              oob_score=False, random_state=7,
                                              verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': range(3, 15),
                         'max_features': range(4, 17),
                         'n_estimators': range(20, 100, 20)},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='recall', verbose=True)
```

In [ ]:
```
model_cv.best_params_
```

Out[ ]:
```
{'max_depth': 14, 'max_features': 13, 'n_estimators': 60}
```

The best parameters found by grid search are max_depth=11, max_features=8 and n_estimators =80

max_depth: maximum depth of the tree

max_features: the number of features to consider for finding best split

n_estimators: the number of trees by the random forest model

In [ ]:
```
model_cv.best_score_
```

Out[ ]:
```
0.7484848484848484
```

In [ ]:
```
model_cv.best_estimator_
```

Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=14, max_features=13,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=60, n_jobs=-1,
                       oob_score=False, random_state=7, verbose=0,
                       warm_start=False)
```

Question-7: Train the Random Forest model

Tasks to do:

> Train the random forest model using the parameters found by Grid Search CV
>
> Also, predict the classes for test data

In [ ]:
```
final_model = model_cv.best_estimator_
final_model.fit(X_train,y_train)
y_pred=final_model.predict(X_test)
```

Question-8: Evaluate the model

Tasks to do:

Evaluate the model using accuracy score

Evaluate the model using classification report

In [ ]:
```
from sklearn.metrics import accuracy_score, classification_report
print(f"Accuracy score on test data is {accuracy_score(y_test,y_pred)}")
print(classification_report(y_test,y_pred))
```

```
Accuracy score on test data is 0.9568480300187617
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       472
           1       0.93      0.67      0.78        61

    accuracy                           0.96       533
   macro avg       0.95      0.83      0.88       533
weighted avg       0.96      0.96      0.95       533
```

Question-9: Find the ten best features and perform cross-validation using just ten features

Tasks to do:

Find the ten best features from the random forest model

Create a new data frame with the ten best features

Perform K-Fold cross-validation on the new dataset with reduced features

Compare the score with previous K-fold cross-validation

In [ ]:
```
from numpy import argsort
imp = final_model.feature_importances_
x= argsort(imp)[::-1]
col_names=list(df.columns)
col_names.remove('State')
col_names.remove('Churn')
top_features = list()
for i in x[:13]:
  print(col_names[i])
  top_features.append(col_names[i])
```

```
Total day charge
Customer service calls
Total eve charge
International plan
Total intl calls
Total intl minutes
avg_eve_call_dur
Total intl charge
avg_day_call_dur
Total night charge
avg_night_call_dur
Number vmail messages
state_churn
```

In [ ]:
```
X_new = df[top_features]
X_new
```

Out[ ]:

	Total day charge	Customer service calls	Total eve charge	International plan	Total intl calls	Total intl minutes	avg_eve_call_dur	Total intl charge	avg_day_call_dur	Total night charge	avg_night_call_dur	Number vmail messages
0	45.07	1	16.78	0	3	10.0	1.993939	2.70	2.410000	11.01	2.689011	25
1	27.47	1	16.62	0	3	13.7	1.898058	3.70	1.313821	11.45	2.469903	26
2	41.38	0	10.30	0	5	12.2	1.101818	3.29	2.135088	7.32	1.563462	0
3	50.90	2	5.26	1	7	6.6	0.703409	1.78	4.216901	8.86	2.212360	0
4	28.34	3	12.61	1	3	10.1	1.215574	2.73	1.475221	8.41	1.544628	0
...	...	...	...	...	...	...	...	...	...	...	...	...
2661	22.90	2	16.12	0	5	11.8	2.789706	3.19	1.374490	9.96	1.729688	0
2662	26.55	2	18.32	0	6	9.9	1.710317	2.67	2.028571	12.56	3.362651	36
2663	39.29	3	13.04	0	4	9.6	2.789091	2.59	4.054386	8.61	1.555285	0
2664	30.74	2	24.55	0	6	14.1	4.979310	3.81	1.658716	8.64	2.108791	0
2665	39.85	0	22.60	0	4	13.7	3.242683	3.70	2.074336	10.86	3.135065	25

2663 rows × 13 columns

In [ ]:
```
new_model = model_cv.best_estimator_
score=cross_val_score(new_model, X_new, y, cv=5)
print(f'score of random forest classifier:{score.mean()}')
```

```
score of random forest classifier:0.9463069023402785
```

We can see with just 10 features accuracy is quite good as compared to previous K-Fold cross validation

> Current Score: 0.946
>
> Previous Score: 0.947

Scenario 2: Chronic Kidney Disease Prediction

The dataset is taken over two months in India. It has 400 rows with 26 features like red blood cells, pedal edema, sugar, etc.

Problem Statement:

You aim to classify whether a patient has chronic kidney disease or not.

Dataset Description:

The dataset contains 26 features:

age - age

bp - blood pressure

sg - specific gravity

al - albumin

su - sugar

rbc - red blood cells

pc - pus cell

pcc - pus cell clumps

ba - bacteria

bgr - blood glucose random

bu - blood urea

sc - serum creatinine

sod - sodium

pot - potassium

hemo - hemoglobin

pcv - packed cell volume

wc - white blood cell count

rc - red blood cell count

htn - hypertension

dm - diabetes mellitus

cad - coronary artery disease

appet - appetite

pe - pedal edema

ane - anemia

class - class

Tasks to be performed:

1. Load the data, check its shape and check for null values, check unique values for categorical feature, Convert features into appropriate data type - Beginner
2. Convert categorical to numerical feature using Label Encoder - Intermediate
3. Plot and analyze Correlation - Beginner
4. Split the dataset for training and testing - Beginner

1. Perform Grid Search cross-validation also perform prediction using the best model - Intermediate (Bridging Question)
2. Evaluate the model using a classification report and confusion matrix - Beginner

Topics Covered:

Data collection

Data analysis

Data wrangling/Feature engineering

Train/Test Algorithms

Perform Grid Search Cross-Validation

Predicting using the trained model

Evaluating a model: Confusion matrix and Classification Report

In [ ]:
```
!wget https://www.dropbox.com/s/hd1jir21dmlgh27/kidney_disease.csv
```

```
--2020-07-20 07:48:37--  https://www.dropbox.com/s/hd1jir21dmlgh27/kidney_disease.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/hd1jir21dmlgh27/kidney_disease.csv [following]
--2020-07-20 07:48:38--  https://www.dropbox.com/s/raw/hd1jir21dmlgh27/kidney_disease.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucb48cbe3708682ca4e69ed1dc18.dl.dropboxusercontent.com/cd/0/inline/A73hM6vWgGy-sUFvhvZnpyCTBLDNx9Jy
s7dRcHK9LeFY9NvkSWhRDkB0-_faL9785b5zK4-Ces8vj4rQGZTdmrQ8Km8xWTMCBKlDIRJnDKyi7f-kNRQPyzIwE7nrPhgr6CM/file# [following]
--2020-07-20 07:48:38--  https://ucb48cbe3708682ca4e69ed1dc18.dl.dropboxusercontent.com/cd/0/inline/A73hM6vWgGy-sUFvh
vZnpyCTBLDNx9Jys7dRcHK9LeFY9NvkSWhRDkB0-_faL9785b5zK4-Ces8vj4rQGZTdmrQ8Km8xWTMCBKlDIRJnDKyi7f-kNRQPyzIwE7nrPhgr6CM/fi
le
Resolving ucb48cbe3708682ca4e69ed1dc18.dl.dropboxusercontent.com (ucb48cbe3708682ca4e69ed1dc18.dl.dropboxusercontent.
com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to ucb48cbe3708682ca4e69ed1dc18.dl.dropboxusercontent.com (ucb48cbe3708682ca4e69ed1dc18.dl.dropboxusercont
ent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48551 (47K) [text/plain]
Saving to: ‘kidney_disease.csv’

kidney_disease.csv  100%[===================>]  47.41K  --.-KB/s    in 0.03s

2020-07-20 07:48:39 (1.82 MB/s) - ‘kidney_disease.csv’ saved [48551/48551]
```

Question-1: Load and analyze the data

Tasks to do:

Load the data in a pandas DataFrame
Have a look at the first five rows
Check if the dataset contains any null values
Check the shape of the dataset
Check the unique values of the categorical columns
Check the data-types of all the columns
Convert features to appropriate data types

In [ ]:
```
import pandas as pd
data = pd.read_csv('/content/kidney_disease.csv')
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(400, 26)
```

In [ ]:
```
data.head()
```

Out[ ]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	a
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4	44	7800	5.2	yes	yes	no	
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	0.8	NaN	NaN	11.3	38	6000	NaN	no	no	no	
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	1.8	NaN	NaN	9.6	31	7500	NaN	no	yes	no	
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	3.8	111.0	2.5	11.2	32	6700	3.9	yes	no	no	
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	1.4	NaN	NaN	11.6	35	7300	4.6	no	no	no	

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              400 non-null    int64  
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object 
 7   pc              335 non-null    object 
 8   pcc             396 non-null    object 
 9   ba              396 non-null    object 
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object 
 17  wc              295 non-null    object 
 18  rc              270 non-null    object 
 19  htn             398 non-null    object 
 20  dm              398 non-null    object 
 21  cad             398 non-null    object 
 22  appet           399 non-null    object 
 23  pe              399 non-null    object 
 24  ane             399 non-null    object 
 25  classification  400 non-null    object 
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

We can see pcv, wc and rc column have numeric values but there dtypes is object. We need to fix this

In [ ]:
```
data.pcc.unique()
```

Out[ ]:
```
array(['notpresent', 'present', nan], dtype=object)
```

In [ ]:
```
clean_df = data.dropna()
```

In [ ]:
```
clean_df.isnull().sum()
```

Out[ ]:
```
id                0
age               0
bp                0
sg                0
al                0
su                0
rbc               0
pc                0
pcc               0
ba                0
bgr               0
bu                0
sc                0
sod               0
pot               0
hemo              0
pcv               0
wc                0
rc                0
htn               0
dm                0
cad               0
appet             0
pe                0
ane               0
classification    0
dtype: int64
```

In [ ]:
```
clean_df.shape
```

Out[ ]:
```
(158, 26)
```

In [ ]:
```
clean_df['pcv']=clean_df[['pcv']].astype(float)
clean_df['wc'] = clean_df['wc'].astype(float)
clean_df['rc'] = clean_df['rc'].astype(float)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Question-2: We cannot use string objects for prediction, so convert the categorical feature to numerical values

Tasks to do:

Convert all the categorical features to numerical values using Label Encoder from sklearn

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in clean_df.columns:
  if clean_df[i].dtype=='object':
    clean_df[i]=le.fit_transform(clean_df[i])
clean_df.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  """
```

Out[ ]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	c
3	3	48.0	70.0	1.005	4.0	0.0	1	0	1	0	117.0	56.0	3.8	111.0	2.5	11.2	32.0	6700.0	3.9	1	0	0	1	1	1	
9	9	53.0	90.0	1.020	2.0	0.0	0	0	1	0	70.0	107.0	7.2	114.0	3.7	9.5	29.0	12100.0	3.7	1	1	0	1	0	1	
11	11	63.0	70.0	1.010	3.0	0.0	0	0	1	0	380.0	60.0	2.7	131.0	4.2	10.8	32.0	4500.0	3.8	1	1	0	1	1	0	
14	14	68.0	80.0	1.010	3.0	2.0	1	0	1	1	157.0	90.0	4.1	130.0	6.4	5.6	16.0	11000.0	2.6	1	1	1	1	1	0	
20	20	61.0	80.0	1.015	2.0	0.0	0	0	0	0	173.0	148.0	3.9	135.0	5.2	7.7	24.0	9200.0	3.2	1	1	1	1	1	1	

Question-3: Plot the correlation and tell which feature will help the most while prediction

Tasks to do:

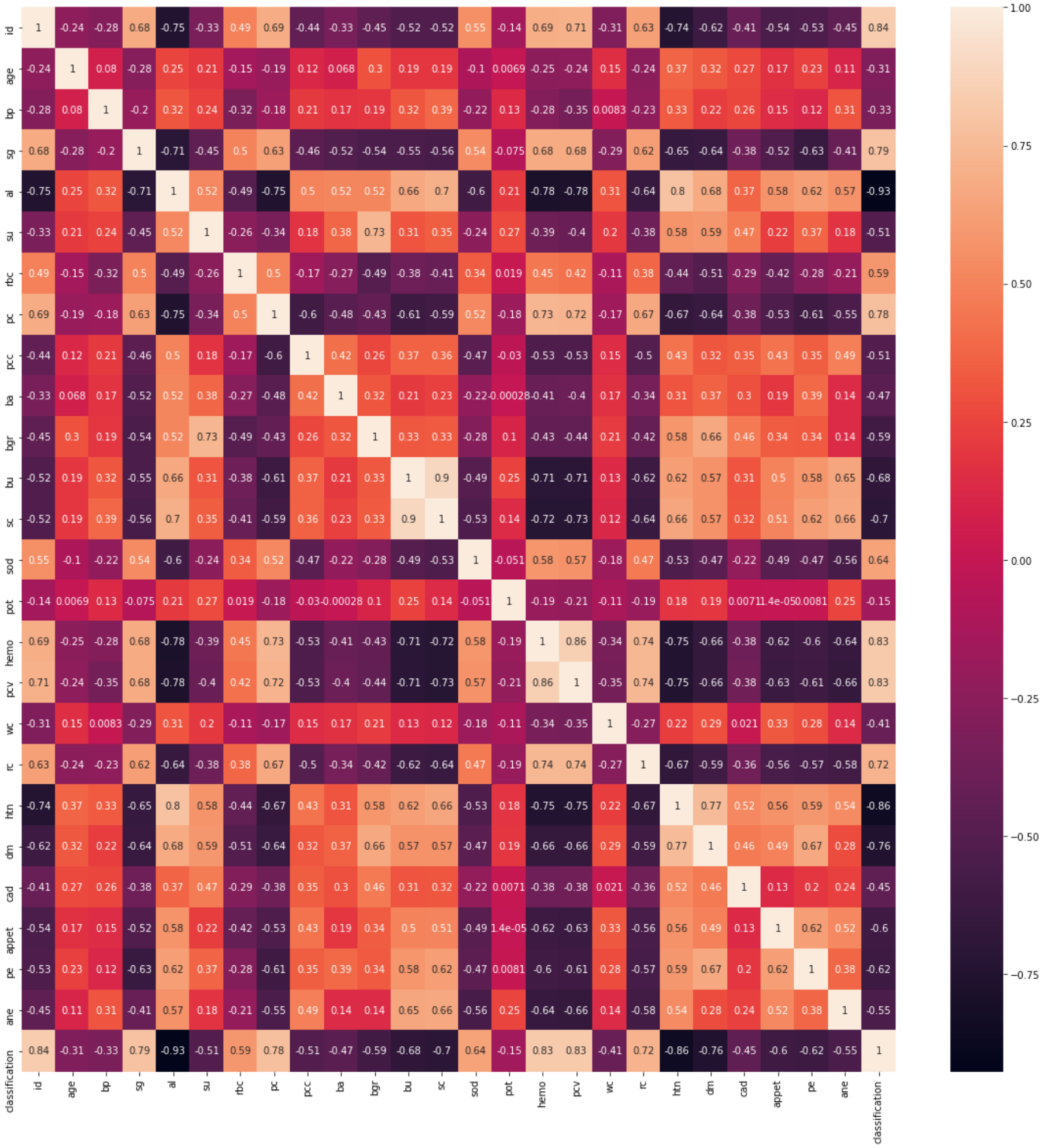
Calculate correlation

Plot the correlation

Compare the correlation

In [ ]:
```
import seaborn as sns
from matplotlib import pyplot as plt
fig, ax = plt.subplots(figsize=(20,20))
correlation = clean_df.corr()
sns.heatmap(correlation,annot=True,ax=ax)
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff804f64198>



We can see each of the 25 independent features are showing quite good correlation with the target variable

Question-4: Split the data into training and testing datasets

Tasks to do:

Split the dataset using sklearn, with 30% for testing with random_state=1

In [ ]:
```
from sklearn.model_selection import train_test_split
X= clean_df.drop(columns=['classification'],axis=1)
y= clean_df['classification']
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.30,random_state = 1)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(110, 25)
(48, 25)
(110,)
(48,)
```

Question-5: Perform Grid Search cross validation to find the best parameters for Decision Tree Classifier

Tasks to do:

Perform Grid Search CV on random forest classifier to find the best number of:

- maximum depth for the trees, between 3 and 25
- maximum features to be used, between range 4 and 22

Cross validate the model with 5 folds

Check the best score and best estimator found by cross validation

Perform prediction using the model found in cross validation

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
model = DecisionTreeClassifier(random_state=7)
model_params = {'criterion':['gini', 'entropy'], 'max_depth':range(3,25), 'max_features':range(4,22)}
model_cv = GridSearchCV(model,model_params,cv=5,n_jobs=-1,verbose=True)
model_cv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 792 candidates, totalling 3960 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 956 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done 3957 out of 3960 | elapsed:   13.0s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done 3960 out of 3960 | elapsed:   13.0s finished
```

Out[ ]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort='deprecated',
                                              random_state=7, splitter='best'),
             iid='deprecated', n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(3, 25),
                         'max_features': range(4, 22)},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=True)
```

In [ ]:
```
model_cv.best_estimator_
```

Out[ ]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=4, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=7, splitter='best')
```

In [ ]:
```
model_cv.best_score_
```

Out[ ]:
```
0.990909090909091
```

In [ ]:
```
model_cv.best_params_
```

Out[ ]:
```
{'criterion': 'gini', 'max_depth': 3, 'max_features': 4}
```

In [ ]:
```
y_pred=model_cv.predict(X_test)
```

Question-6: Evaluate the model

Tasks to do:

Evaluate the model using confusion matrix

Evaluate the model using classification report

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[13  0]
 [ 0 35]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00        35

    accuracy                           1.00        48
   macro avg       1.00      1.00      1.00        48
weighted avg       1.00      1.00      1.00        48
```

In [ ]:
```

```

Module-4: Mathematical and Bayesian Models

Classification is the process of classifying data in the different categories based on training data. In this question bank, we will be apply Naive Bayes, SVM, and KNN to classify the data.

Getting Datasets:

In [ ]:
```
!wget https://www.dropbox.com/s/mcflbukhp33tbj2/plastics_type.csv -nv
!wget https://www.dropbox.com/s/rtkd1f4iulf8jp0/auto-mpg.csv -nv
!wget https://www.dropbox.com/s/oqsri5a5a85tlaa/guest_data.csv -nv
```

```
2020-07-14 04:35:06 URL:https://ucf8f5a4b020d8e78cd57cbee273.dl.dropboxusercontent.com/cd/0/inline/A7ekratMtZP9ZLod-w
5bxcBvZoLsbQ3oOVLYZg4Mp4wICYtlkqbw0Mk-xPsIXf7lrBJelV6IHaTfiKqtTSNjiRFrCheIVqeUn040W9j7DWON-JFJ-8ZmNskz7RmaCyVXhT4/fil
e [4359/4359] -> "plastics_type.csv" [1]
2020-07-14 04:35:09 URL:https://uccd567702e1ed57238ee204c8a8.dl.dropboxusercontent.com/cd/0/inline/A7cjBSuflKE2eFNklg
FWrPA7fgFPUeEDo2yHltdA1LZApp9BMGL88TidwkZmUr30A0E1dG1-E9hfVP1kEmPZ5ciyqVO7Hm1IlgbxvDxJS6HFeqSDLqHm8peg_YqJBNvAW2g/fil
e [19677/19677] -> "auto-mpg.csv" [1]
2020-07-14 04:35:12 URL:https://uc6980d7aa353f88482018558b49.dl.dropboxusercontent.com/cd/0/inline/A7cSPxljM9fxJGHAlb
tccdXMFgxPFe9Pem-U4M6zG8xtgLn-bCCOdazd0H13V6se_D607WQotTePFAWCLMY48DIBZNsgbCI4FdrXTKPrGGEGjkl539E6KtOk3f3UtNW7m9U/fil
e [5319743/5319743] -> "guest_data.csv" [1]
```

Scenario 1: Emerald Oyster Resorts

Emerald Oyster is proud to have been catering business and luxury travelers for more than 50 years. The comprehensive offering of high-end hotel stays customer services, and with best in class chefs of every locus. The accommodations are designed in such a way that they reflect the surrounding environment with decor and hotel's location. Each hotel exhibits its sense of style, individual properties with 24-hour customer service.

Problem Statement:

Recent tourists say you can stick to the high cost of meals in most restaurants in the resort. The amenities can be remodeled with a new menu and choices based on customer economic condition to resolve this issue. As the resorts cover a wide area, only rich people go in depths of the resorts who buy the shuttle services; customers with a low budget only explore the outer parts of the resorts. The task is to identify the customer economic class based on the data collected so that the board can resolve this with proper remodeling.

Tasks to be Performed:

In order to attain the above goal below, tasks must be performed:

- Read the dataset with no headers; put respective columns names, and find the missing values in each column. - **Beginner**
- Process any null values present in the data with the respective method. - **Beginner**
- Create two datasets, one using a dummy variable and one using LabelEncoder. - **Intermediate**
- Split both the datasets into training and testing set and apply Gaussian Naive Bayes. - **Intermediate**
- Evaluate both the model using a confusion matrix. - **Advanced**
- Apply Bernoulli Naive Bayes on both the datasets and re-evaluate them. - **Advanced**

Dataset Description:

The data collected has 45222 observations and 14 features along with the target variable *Class* which identifies the income of the individual.

Features:

- **Age**: Real [17.0, 90.0]
- **Workclass**: {Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked}
- **Fnlwgt**: Real [12285.0, 1490400.0]
- **Education**: {Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool}
- **Education-num**: Real [1.0, 16.0]
- **Marital-status**: {Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse}
- **Occupation**: {Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces}
- **Relationship**: {Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried}
- **Race**: {White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black}
- **Sex**: {Female, Male}
- **Capital-gain**: Real [0.0, 99999.0]
- **Capital-loss**: Real [0.0, 4356.0]
- **Hours-per-week**: Real [1.0, 99.0]
- **Native-country**: {United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands}

Target Variable:

- **Class**: {>50K, <=50K}

Topics Covered:

- Naive Bayes
- Confusion Matrix

Question-1: Read the dataset with no headers; put respective columns names and find the missing values in each column.

In [ ]:
```
import pandas as pd
```

In [ ]:
```
cols=['Age','Workaclass','Fnlwgt','Education','Edu_num',
      'Marital_Status','Occupation','Relationship','Race','Sex','Capital_gain','Capital_loss',
     'HPW','Native_Country','Target']
```

In [ ]:
```
data=pd.read_csv('guest_data.csv',header=None)
data.columns=cols
```

In [ ]:
```
data.head()
```

Out[ ]:

	Age	Workaclass	Fnlwgt	Education	Edu_num	Marital_Status	Occupation	Relationship	Race	Sex	Capital_gain	Capital_loss	HPW	Nat
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	U
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	U
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	U
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	U
4	18	NaN	103497	Some-college	10	Never-married	NaN	Own-child	White	Female	0	0	30	U

Basic Data Information

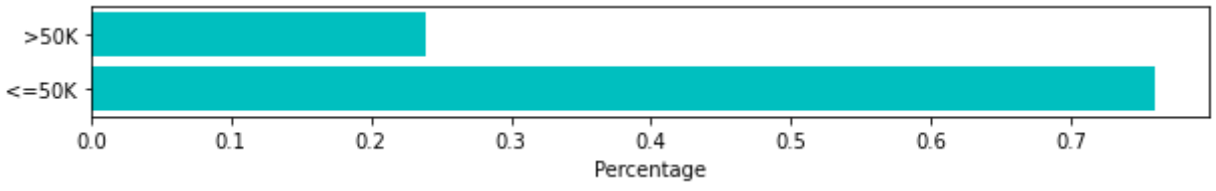
In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Age             48842 non-null  int64 
 1   Workaclass      46043 non-null  object
 2   Fnlwgt          48842 non-null  int64 
 3   Education       48842 non-null  object
 4   Edu_num         48842 non-null  int64 
 5   Marital_Status  48842 non-null  object
 6   Occupation      46033 non-null  object
 7   Relationship    48842 non-null  object
 8   Race            48842 non-null  object
 9   Sex             48842 non-null  object
 10  Capital_gain    48842 non-null  int64 
 11  Capital_loss    48842 non-null  int64 
 12  HPW             48842 non-null  int64 
 13  Native_Country  47985 non-null  object
 14  Target          48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Dataset has 48842 entries.

In [ ]:
```
# Target distribution
def target_distribution(data):
    target_ratio=pd.DataFrame({'Counts':data.Target.value_counts(),
                               'Percentage':data.Target.value_counts()/len(data)})
    import matplotlib.pyplot as plt
    import seaborn as sns
    plt.figure(figsize = (10,1))
    plt.barh(target_ratio.index, target_ratio.Percentage,color='c')
    plt.xlabel('Percentage')
    plt.show()
target_distribution(data)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
ated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```



Here, we can observe that data distribution is 25% and 75% respectively

Question-2: Process any null values present in the data with the respective method.

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
6	Occupation	True	2809	0.057512
1	Workaclass	True	2799	0.057307
13	Native_Country	True	857	0.017546
0	Age	False	0	0.000000
2	Fnlwgt	False	0	0.000000
3	Education	False	0	0.000000
4	Edu_num	False	0	0.000000
5	Marital_Status	False	0	0.000000
7	Relationship	False	0	0.000000
8	Race	False	0	0.000000
9	Sex	False	0	0.000000
10	Capital_gain	False	0	0.000000
11	Capital_loss	False	0	0.000000
12	HPW	False	0	0.000000
14	Target	False	0	0.000000

We can observe that some columns have null values which has to treated by filling in with mode of that respective column

In [ ]:
```
print('Total Missing values: %s'%sum(miss.Count_))
```

```
Total Missing values: 6465
```

In [ ]:
```
import numpy as np
```

In [ ]:
```
for i in miss[miss.Count_!=0].Col_name:
    temp=data[i].mode()
    tes=data[i].isnull()
    ind=tes[tes==True].index
    quill=list(data[i])
    for j in ind:
        quill[j]=temp[0]
    data[i]=quill
```

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?': [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns]})
```

In [ ]:
```
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_
0	Age	False	0
1	Workaclass	False	0
2	Fnlwgt	False	0
3	Education	False	0
4	Edu_num	False	0
5	Marital_Status	False	0
6	Occupation	False	0
7	Relationship	False	0
8	Race	False	0
9	Sex	False	0
10	Capital_gain	False	0
11	Capital_loss	False	0
12	HPW	False	0
13	Native_Country	False	0
14	Target	False	0

Question-3: Create two datasets one using dummy variable and one using labelencoder.

In [ ]:
```
dummy_data=data.copy()
label_data=data.copy()
```

- Dummy Dataset

In [ ]:
```
# findUnique returns a dataframe with every column name, number of unique values, minimum values, maximum values
# and total unique values present altogether in the dataset
def findUnique(data):
    unq=0
    unq_data={'Col_Name':[],'Unique_Counts':[],'Max_value':[],'Min_value':[]}
    for i in data.columns:
        unq_data['Col_Name'].append(i)
        unq_data['Unique_Counts'].append(len(dummy_data[i].unique()))
        unq_data['Max_value'].append(dummy_data[i].max())
        unq_data['Min_value'].append(dummy_data[i].min())
        unq+=len(dummy_data[i].unique())
    return pd.DataFrame(unq_data),unq
u,ud=findUnique(dummy_data)
u.sort_values(by='Unique_Counts')
```

Out[ ]:

	Col_Name	Unique_Counts	Max_value	Min_value
9	Sex	2	Male	Female
14	Target	2	>50K	<=50K
8	Race	5	White	Amer-Indian-Eskimo
7	Relationship	6	Wife	Husband
5	Marital_Status	7	Widowed	Divorced
1	Workaclass	8	Without-pay	Federal-gov
6	Occupation	14	Transport-moving	Adm-clerical
3	Education	16	Some-college	10th
4	Edu_num	16	16	1
13	Native_Country	41	Yugoslavia	Cambodia
0	Age	74	90	17
12	HPW	96	99	1
11	Capital_loss	99	4356	0
10	Capital_gain	123	99999	0
2	Fnlwgt	28523	1490400	12285

In [ ]:
```
print('Total unique values:',ud)
```

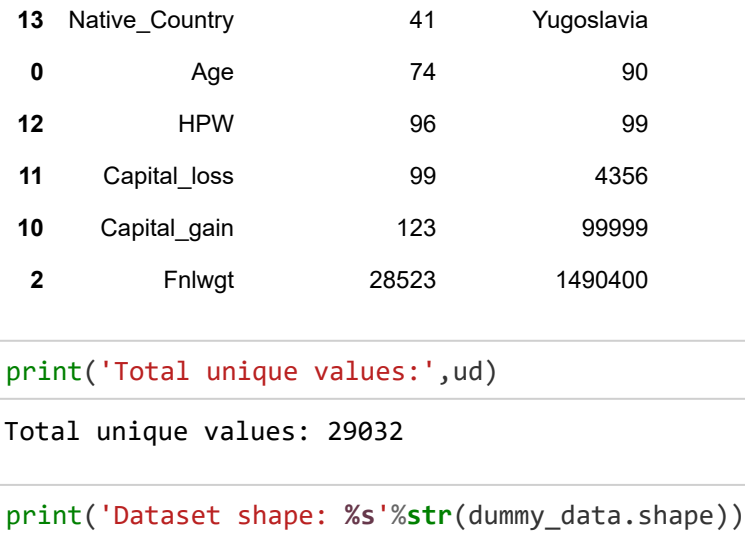
```
Total unique values: 29032
```

In [ ]:
```
print('Dataset shape: %s'%str(dummy_data.shape))
```

```
Dataset shape: (48842, 15)
```

In [ ]:
```
# Since Age and Education are ordinal in nature we should use label encoder on these columns
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
dummy_data.Age=lb.fit_transform(dummy_data.Age)
dummy_data.Education=lb.fit_transform(dummy_data.Education)
dummy_data.HPW=lb.fit_transform(dummy_data.HPW)
```

In [ ]:
```
# All that left is 3 columns 'Capital_gain','Capital_loss' and 'Fnlwgt'
import matplotlib.pyplot as plt
dummy_data.boxplot(column=['Capital_gain','Capital_loss','Fnlwgt'])
plt.show()
```



In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [ ]:
```
scl=MinMaxScaler()
```

In [ ]:
```
dummy_data.Capital_gain=scl.fit_transform(np.array(dummy_data.Capital_gain.values.astype(float)).reshape(1,-1))[0]
dummy_data.Capital_loss=scl.fit_transform(np.array(dummy_data.Capital_loss.values.astype(float)).reshape(1,-1))[0]
dummy_data.Fnlwgt=scl.fit_transform(np.array(dummy_data.Fnlwgt.values.astype(float)).reshape(1,-1))[0]
```

In [ ]:
```
dols=['Workaclass','Edu_num',
      'Marital_Status','Occupation','Relationship','Race','Sex','Native_Country']
dummy_data=pd.get_dummies(dummy_data,columns=dols)
```

In [ ]:
```
print('Dummy Dataset shape: %s'%str(dummy_data.shape))
```

```
Dummy Dataset shape: (48842, 106)
```

In [ ]:
```
dummy_data.head()
```

Out[ ]:

	Age	Fnlwgt	Education	Capital_gain	Capital_loss	HPW	Target	Workaclass_Federal-gov	Workaclass_Local-gov	Workaclass_Never-worked	Workaclass_Priv
0	8	0.0	1	0.0	0.0	39	<=50K	0	0	0	
1	21	0.0	11	0.0	0.0	49	<=50K	0	0	0	
2	11	0.0	7	0.0	0.0	39	>50K	0	1	0	
3	27	0.0	15	0.0	0.0	39	>50K	0	0	0	
4	1	0.0	15	0.0	0.0	29	<=50K	0	0	0	

5 rows × 106 columns

LabelEncode the target variable

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
lb=LabelEncoder()
```

In [ ]:
```
dummy_data.Target=lb.fit_transform(dummy_data.Target)
```

In [ ]:
```
dummy_data.head()
```

Out[ ]:

	Age	Fnlwgt	Education	Capital_gain	Capital_loss	HPW	Target	Workaclass_Federal-gov	Workaclass_Local-gov	Workaclass_Never-worked	Workaclass_Priv
0	8	0.0	1	0.0	0.0	39	0	0	0	0	
1	21	0.0	11	0.0	0.0	49	0	0	0	0	
2	11	0.0	7	0.0	0.0	39	1	0	1	0	
3	27	0.0	15	0.0	0.0	39	1	0	0	0	
4	1	0.0	15	0.0	0.0	29	0	0	0	0	

5 rows × 106 columns

- Label Data

In [ ]:
```
def lb_encode(x):
    lb=LabelEncoder()
    return lb.fit_transform(x)
```

In [ ]:
```
label_data=label_data.apply(lb_encode)
```

In [ ]:
```
label_data.head()
```

Out[ ]:

	Age	Workaclass	Fnlwgt	Education	Edu_num	Marital_Status	Occupation	Relationship	Race	Sex	Capital_gain	Capital_loss	HPW	Native_
0	8	3	19329	1	6	4	6	3	2	1	0	0	39	
1	21	3	4212	11	8	2	4	0	4	1	0	0	49	
2	11	1	25340	7	11	2	10	0	4	1	0	0	39	
3	27	3	11201	15	9	2	6	0	2	1	98	0	39	
4	1	3	5411	15	9	4	9	3	4	0	0	0	29	

Question-4: Split both the datasets into training and testing set and apply Gaussian Naive Bayes.

In [ ]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
```

- Dummy Data

In [ ]:
```
X=dummy_data.drop('Target',axis=1)
y=dummy_data.Target
X_drain, X_dest, y_drain, y_dest = train_test_split(X, y, train_size=0.7,test_size=0.3, random_state=101)
```

In [ ]:
```
dummy_gnb=GaussianNB()
dummy_gnb.fit(X_drain,y_drain)
```

Out[ ]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [ ]:
```
dummy_pred=dummy_gnb.predict(X_dest)
```

- Label Data

In [ ]:
```
lX=label_data.drop('Target',axis=1)
ly=label_data.Target
X_train, X_test, y_train, y_test = train_test_split(lX, ly, train_size=0.7,test_size=0.3, random_state=101)
```

In [ ]:
```
label_gnb=GaussianNB()
label_gnb.fit(X_train,y_train)
```

Out[ ]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [ ]:
```
label_pred=label_gnb.predict(X_test)
```

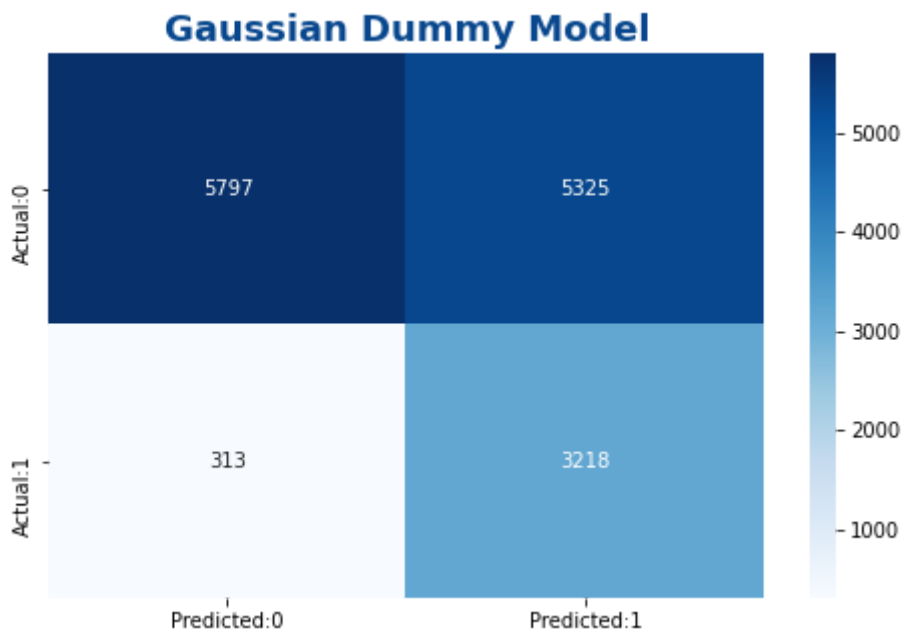
Question-5: Evaluate both the model using confusion matrix.

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score
from sklearn.metrics import classification_report,f1_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
print('The accuracy of the Gaussian dummy model is: ',round(accuracy_score(y_dest,dummy_pred)*100,2))
print('The accuracy of the Gaussian label model is: ',round(accuracy_score(y_test,label_pred)*100,2))
```

```
The accuracy of the Gaussian dummy model is:  61.52
The accuracy of the Gaussian label model is:  81.61
```

In [ ]:
```
dcm=confusion_matrix(y_dest,dummy_pred)
dconf_matrix=pd.DataFrame(data=dcm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(dconf_matrix, annot=True,fmt='d',cmap='Blues')
fm={'size':18,'color':'#08478d','weight':'bold'}
plt.title('Gaussian Dummy Model',**fm)
plt.show()
```



In [ ]:
```
TN=dcm[0,0]
TP=dcm[1,1]
FN=dcm[1,0]
FP=dcm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 5797
True Positive 3218
False Negative 313
False Positive 5325
Sensitivity 0.9113565562163694
Specificity 0.5212192051789246
```

The model is more sensitive than specific means it predicts target *>50K* more accurately.

- **Precision**: Ability of model to distinguish between negative and positive labels.
- **Recall**: Ability of model to find all positive samples.
- **F1 Score**: Weighted harmonic mean of the precision and recall.

In [ ]:
```
# Let's see precision and recall values
#Initializae the evaluation dictionary
def initialize_evaluator():
    return {'Model':[],'Accuracy':[],'Precision':[],'Recall':[],'F1_score':[]}

#Insert data in evaluation dictionary
def insert_data(test,pred,model):
    eval_data=initialize_evaluator()
    eval_data['Model'].append(model)
    eval_data['Accuracy'].append(accuracy_score(test,pred))
    eval_data['Precision'].append(precision_score(test,pred))
    eval_data['Recall'].append(recall_score(test,pred))
    eval_data['F1_score'].append(f1_score(test,pred))
    return eval_data

# Append data of one dictionary to another
def append_data(data1, data2):
    for i in data1.keys():
        data2[i].extend(data1[i])
    return data2

dummy_eval_data=insert_data(y_test,dummy_pred,'Gaussian Dummy Model')
```

In [ ]:
```
pd.DataFrame(dummy_eval_data)
```

Out[ ]:

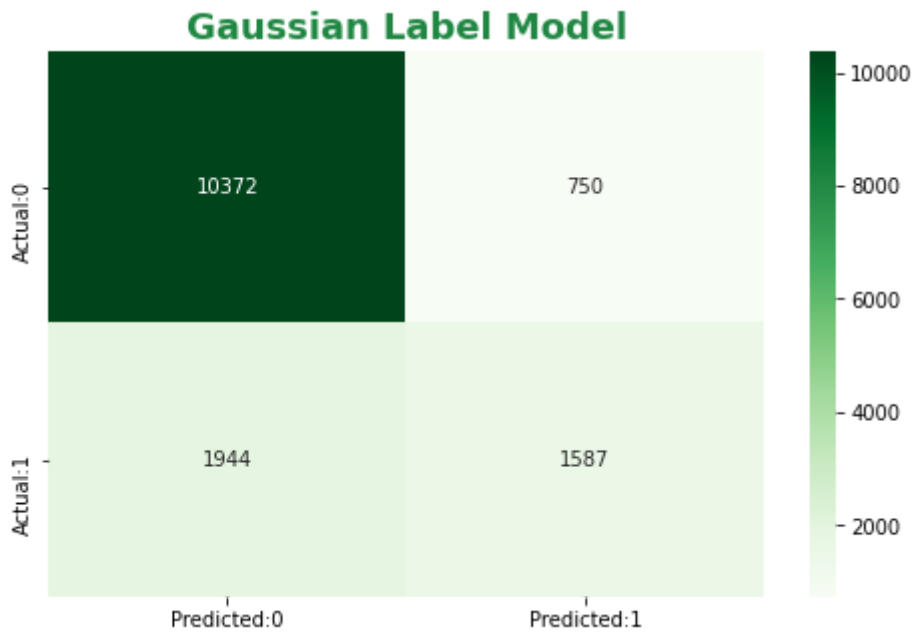
	Model	Accuracy	Precision	Recall	F1_score
0	Gaussian Dummy Model	0.615232	0.376683	0.911357	0.533046

From above we can say that our model is able to correctly classify around 91% guests with >=50K.

In [ ]:
```
# Let's Check the classification report of the model
print(classification_report(y_test,dummy_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.52      0.67     11122
           1       0.38      0.91      0.53      3531

    accuracy                           0.62     14653
   macro avg       0.66      0.72      0.60     14653
weighted avg       0.81      0.62      0.64     14653
```

In [ ]:
```
lcm=confusion_matrix(y_test,label_pred)
lconf_matrix=pd.DataFrame(data=lcm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(lconf_matrix, annot=True,fmt='d',cmap='Greens')
fm={'size':18,'color':'#1f8742','weight':'bold'}
plt.title('Gaussian Label Model',**fm)
plt.show()
```



In [ ]:
```
TN=lcm[0,0]
TP=lcm[1,1]
FN=lcm[1,0]
FP=lcm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 10372
True Positive 1587
False Negative 1944
False Positive 750
Sensitivity 0.4494477485131691
Specificity 0.9325660852364682
```

The model is more specific than sensitive means it predicts target *<50K* more accurately.

In [ ]:
```
label_eval_data=insert_data(y_test,label_pred,'Gaussian Label Model')
pd.DataFrame(label_eval_data)
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1_score
0	Gaussian Label Model	0.816147	0.679076	0.449448	0.5409

While label model is only able to classify around 44.95% of guests >50K.

In [ ]:
```
# Let's Check the classification report of the model
print(classification_report(y_test,label_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.93      0.89     11122
           1       0.68      0.45      0.54      3531

    accuracy                           0.82     14653
   macro avg       0.76      0.69      0.71     14653
weighted avg       0.80      0.82      0.80     14653
```

In [ ]:
```
eval_data=append_data(label_eval_data,dummy_eval_data)
```

Question-6: Apply Bernoulli Naive Bayes on both the datasets and evaluate them again.

In [ ]:
```
from sklearn.naive_bayes import BernoulliNB
```

- Dummy Data

In [ ]:
```
dummy_gnb=BernoulliNB(alpha=2)
dummy_gnb.fit(X_drain,y_drain)
```

Out[ ]:
```
BernoulliNB(alpha=2, binarize=0.0, class_prior=None, fit_prior=True)
```

In [ ]:
```
dummy_pred=dummy_gnb.predict(X_dest)
```

- Label Data

In [ ]:
```
label_gnb=BernoulliNB(alpha=2)
label_gnb.fit(X_train,y_train)
```

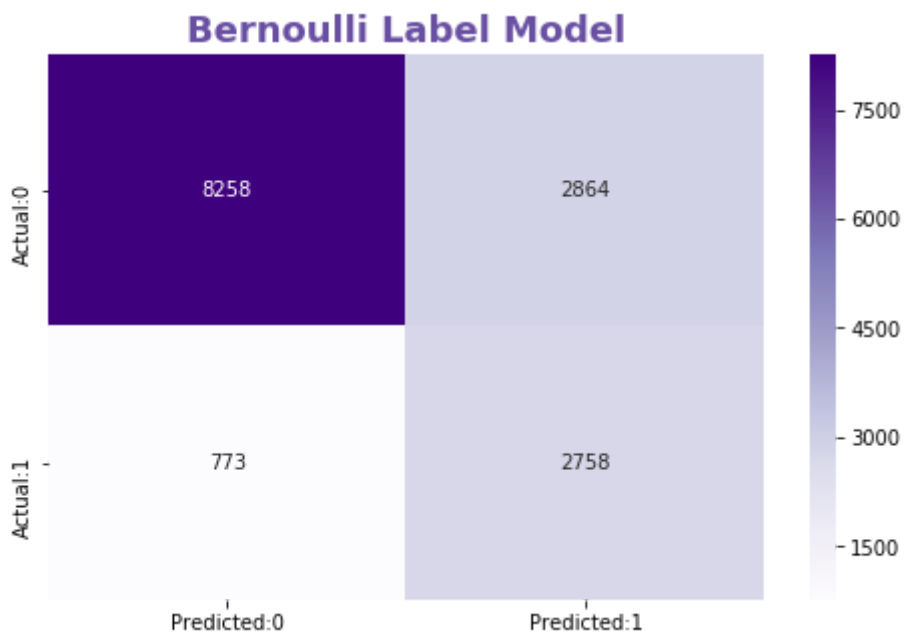
Out[ ]:
```
BernoulliNB(alpha=2, binarize=0.0, class_prior=None, fit_prior=True)
```

In [ ]:
```
label_pred=label_gnb.predict(X_test)
```

In [ ]:
```
print('The accuracy of the bernoulli dummy model is: ',round(accuracy_score(y_dest,dummy_pred)*100,2))
print('The accuracy of the bernoulli label model is: ',round(accuracy_score(y_test,label_pred)*100,2))
```

```
The accuracy of the bernoulli dummy model is:  75.18
The accuracy of the bernoulli label model is:  73.41
```

In [ ]:
```
dcm=confusion_matrix(y_dest,dummy_pred)
dconf_matrix=pd.DataFrame(data=dcm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(dconf_matrix, annot=True,fmt='d',cmap='Purples')
fm={'size':18,'color':'#694fa2','weight':'bold'}
plt.title('Bernoulli Label Model',**fm)
plt.show()
```



In [ ]:
```
TN=dcm[0,0]
TP=dcm[1,1]
FN=dcm[1,0]
FP=dcm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 5797
True Positive 3218
False Negative 313
False Positive 5325
Sensitivity 0.9113565562163694
Specificity 0.5212192051789246
```

In [ ]:
```
bern_dummy_eata=insert_data(y_test,dummy_pred,'Bernoulli Dummy Model')
pd.DataFrame(bern_dummy_eata)
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1_score
0	Bernoulli Dummy Model	0.751791	0.490573	0.781082	0.602644

The model is able to correctly classify 78% of positive data i.e. >=50K.

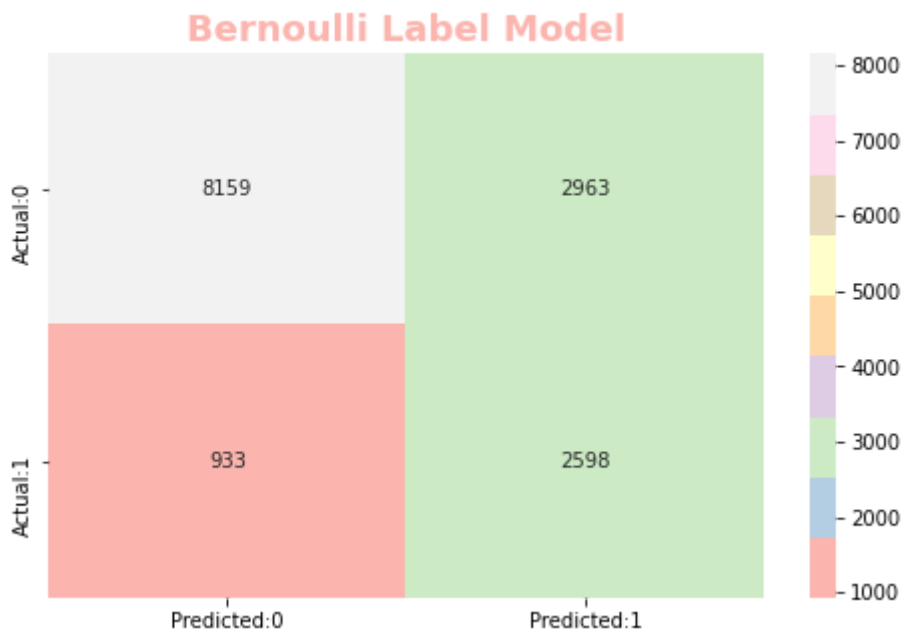
In [ ]:
```
print(classification_report(y_test,dummy_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.74      0.82     11122
           1       0.49      0.78      0.60      3531

    accuracy                           0.75     14653
   macro avg       0.70      0.76      0.71     14653
weighted avg       0.81      0.75      0.77     14653
```

In [ ]:
```
eval_data=append_data(bern_dummy_eata,eval_data)
```

The model is highly specific than sensitive means it predicts target *<50K* more accurately.

In [ ]:
```
lcm=confusion_matrix(y_test,label_pred)
lconf_matrix=pd.DataFrame(data=lcm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(lconf_matrix, annot=True,fmt='d',cmap='Pastel1')
fm={'size':18,'color':'#fbb4ae','weight':'bold'}
plt.title('Bernoulli Label Model',**fm)
plt.show()
```



In [ ]:
```
TN=lcm[0,0]
TP=lcm[1,1]
FN=lcm[1,0]
FP=lcm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 8159
True Positive 2598
False Negative 933
False Positive 2963
Sensitivity 0.735768903993203
Specificity 0.733591080740874
```

The model is both specific and sensitive.

In [ ]:
```
bern_label_model=insert_data(y_test,label_pred,'Bernoulli Label Model')
pd.DataFrame(bern_label_model)
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1_score
0	Bernoulli Label Model	0.734116	0.467182	0.735769	0.571491

In [ ]:
```
eval_data=append_data(bern_label_model,eval_data)
```

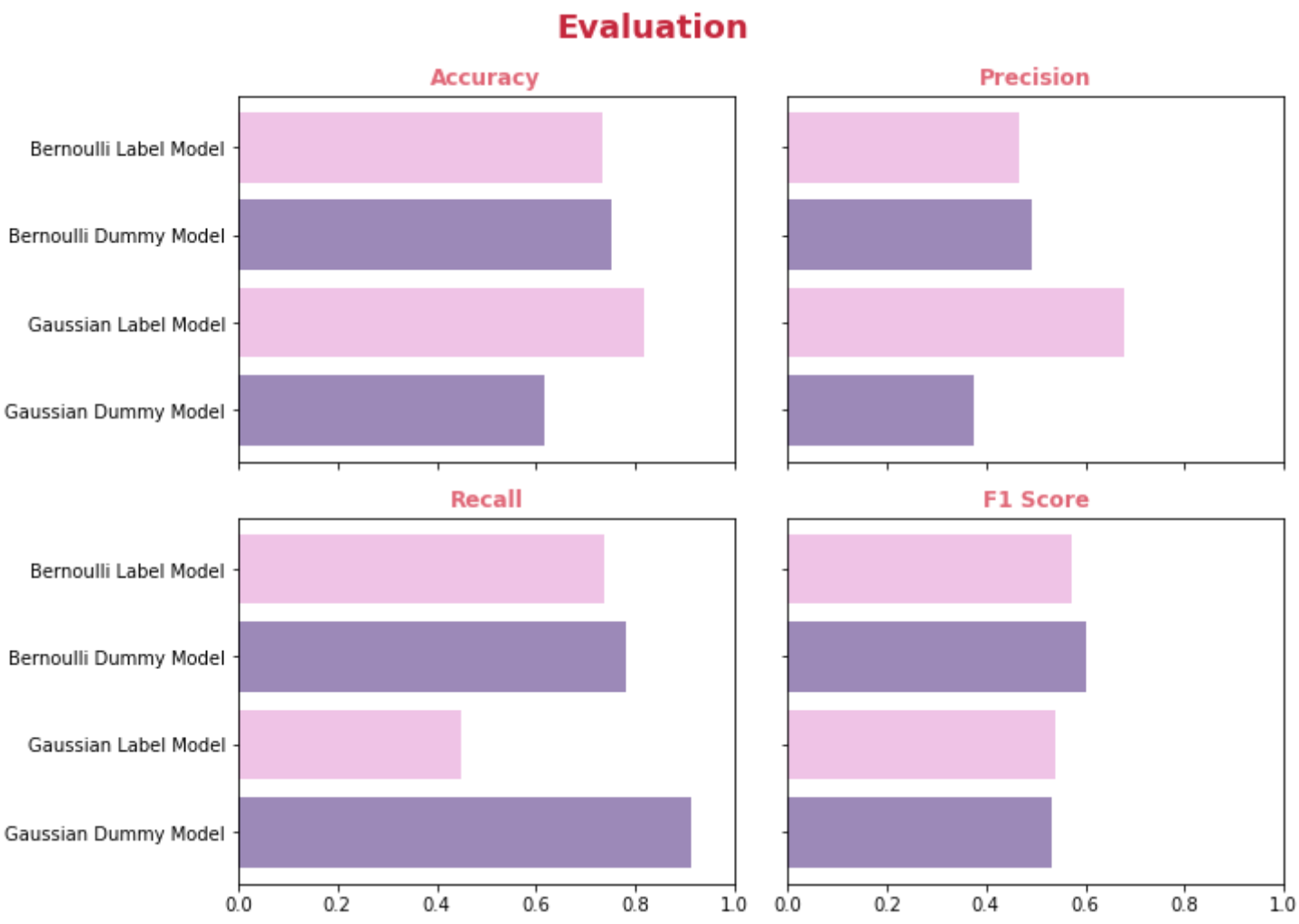
In [ ]:
```
pd.DataFrame(eval_data)
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1_score
0	Gaussian Dummy Model	0.615232	0.376683	0.911357	0.533046
1	Gaussian Label Model	0.816147	0.679076	0.449448	0.540900
2	Bernoulli Dummy Model	0.751791	0.490573	0.781082	0.602644
3	Bernoulli Dummy Model	0.751791	0.490573	0.781082	0.602644
4	Bernoulli Dummy Model	0.751791	0.490573	0.781082	0.602644
5	Bernoulli Label Model	0.734116	0.467182	0.735769	0.571491

In [ ]:
```
from matplotlib.gridspec import GridSpec
def plot_models(data):
    sns.set_palette(sns.color_palette("rocket"))
    super_title={'size':18,'color':'#c5283d','weight':'bold'}
    sub_title={'size':12,'color':'#e06777','weight':'bold'}
    colors=np.array([[156, 137, 184],[239, 195, 230],[184, 190, 221],[231, 115, 171]])
    colors=colors/255 #Matplotlib RGB color range is from 0-1
    data=pd.DataFrame(data)
    fig = plt.figure(figsize=(10,7),constrained_layout=True)
    gs = GridSpec(2, 2, figure=fig)
    ax1 = fig.add_subplot(gs[0, 0])
    ax1.barh(data.Model,data.Accuracy,color=colors)
    ax1.tick_params(labelbottom=False, labelleft=True)
    ax1.set_xlim(0,1)
    ax1.set_title('Accuracy',**sub_title)
    ax2 = fig.add_subplot(gs[0, 1])
    ax2.barh(data.Model,data.Precision,color=colors)
    ax2.tick_params(labelbottom=False, labelleft=False)
    ax2.set_xlim(0,1)
    ax2.set_title('Precision',**sub_title)
    ax3 = fig.add_subplot(gs[1, 0])
    ax3.barh(data.Model,data.Recall,color=colors)
    ax3.tick_params(labelbottom=True, labelleft=True)
    ax3.set_xlim(0,1)
    ax3.set_title('Recall',**sub_title)
    ax4 = fig.add_subplot(gs[1, 1])
    ax4.barh(data.Model,data.F1_score,color=colors)
    ax4.tick_params(labelbottom=False, labelleft=False)
    ax4.set_xlim(0,1)
    ax4.set_title('F1 Score',**sub_title)
    fig.suptitle("Evaluation",**super_title)
    ax4.tick_params(labelbottom=True, labelleft=False)
    plt.show()
```

In [ ]:
```
plot_models(eval_data)
```



From the above graphs, we can determine the best model. Overall, Gaussian Dummy Model has the best recall, but accuracy is pretty low as well as precision score. Even Gaussian Label Model has the best accuracy, but the truth is that the model performs poorly as the recall is worst. On average sense, we can see that our Bernoulli Label model is performing well enough to be selected.

Conclusion: Even though the dummy Bernoulli model produced better accuracy, it would not be appreciated if it used as it is highly specific. As the Bernoulli label model has a good accuracy of 73.41%, it would best if this is used. As discussed in the module Gaussian Naive Bayes model is always best used when data is continuous and normally distributed, which is why dummy Gaussian naive Bayes failed to produce better accuracy than label gaussian model. Bernoulli model is best used when we have independent features and have a binary outcome.

Scenario-2: Gnar Automobiles

Gnar Automobiles engages in the distribution and sale of automobiles and light commercial vehicles. The owner of the Gnar Automobiles deals with a number of distributors across countries in different origins.

Problem Statement

As every origin sends cars with various specifications. The owner wants to determine the origin of the cars based on the specifications of the cars to further increase business opportunities.

Dataset Description

"The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." Attribute Information:

- **mpg**: continuous
- **cylinders**: multi-valued discrete
- **displacement**: continuous
- **horsepower**: continuous
- **weight**: continuous
- **acceleration**: continuous
- **model year**: multi-valued discrete
- **origin**: multi-valued discrete
- **car name**: string (unique for each instance)

Tasks to be Performed:

In order to attain the above goal below, tasks must be performed:

- Read the dataset and process the missing values in each column. - **Beginner**
- Scale the data using MinMaxScaler and plot the boxplot before and after scaling the data. - **Beginner**
- Explore the data relations with the target variable *origin*. - **Intermediate**
- Split the data into training and testing set and apply the KNN model with k=3,9,12. - **Intermediate**
- Evaluate the model and find the accuracy score. - **Intermediate**
- Plot the visualizations for the models. - **Advanced**

Topics Covered:

- K-Nearest Neighbor

Importing Required Libraries

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from mlxtend.plotting import plot_decision_regions
%matplotlib inline
```

Question-1: Read the dataset and process the missing values in each column.

In [ ]:
```
data=pd.read_csv('auto-mpg.csv')
```

In [ ]:
```
data.head()
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

In [ ]:
```
data.describe(include='all')
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000	398.000000	398
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	305
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	ford pinto
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050	1.572864	NaN
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627	0.802055	NaN
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000	NaN
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000	1.000000	NaN
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000	1.000000	NaN
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000	2.000000	NaN
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000	NaN

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
3	horsepower	True	6	0.015075
0	mpg	False	0	0.000000
1	cylinders	False	0	0.000000
2	displacement	False	0	0.000000
4	weight	False	0	0.000000
5	acceleration	False	0	0.000000
6	model year	False	0	0.000000
7	origin	False	0	0.000000
8	car name	False	0	0.000000

In [ ]:
```
print('Total Missing values: %s'%sum(miss.Count_))
```

```
Total Missing values: 6
```

In [ ]:
```
data.horsepower=data.horsepower.fillna(data.horsepower.mean())
```

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

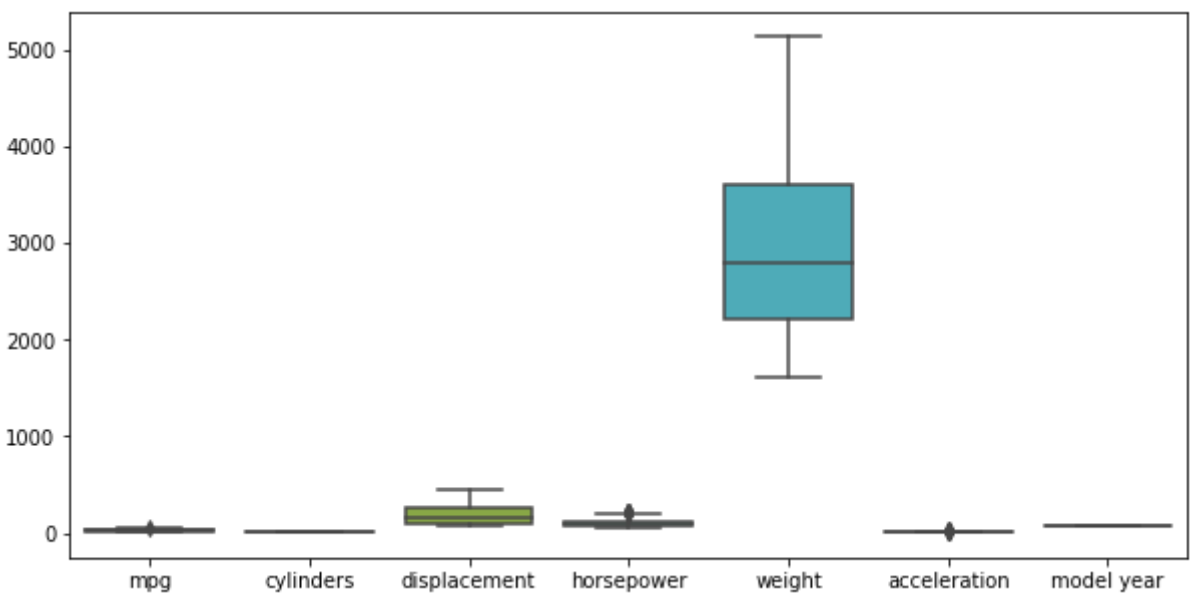
	Col_name	Missing value?	Count_	Percentage
0	mpg	False	0	0.0
1	cylinders	False	0	0.0
2	displacement	False	0	0.0
3	horsepower	False	0	0.0
4	weight	False	0	0.0
5	acceleration	False	0	0.0
6	model year	False	0	0.0
7	origin	False	0	0.0
8	car name	False	0	0.0

Question-2: Scale the data using MinMaxScaler and plot the boxplot before and after scaling the data.

In [ ]:
```
feat=data.columns
feat=feat.drop(['car name', 'origin'])
```

In [ ]:
```
X=data.drop(['car name', 'origin'],axis=1)
y=data.origin
```

In [ ]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=X)
plt.xticks(ticks=np.arange(len(feat)),labels=feat)
plt.show()
```

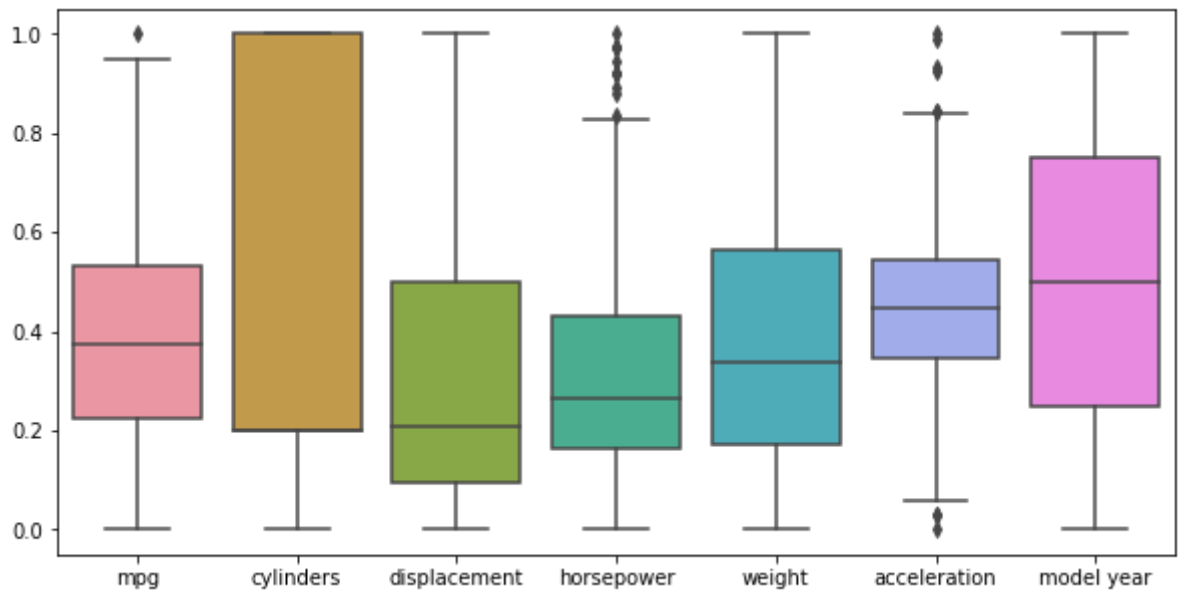


The data is highly imbalanced. Using MinMaxScaler to scale the data.

In [ ]:
```
scl=MinMaxScaler()
```

In [ ]:
```
X=scl.fit_transform(X)
```

In [ ]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=X)
plt.xticks(ticks=np.arange(len(feat)),labels=feat)
plt.show()
```

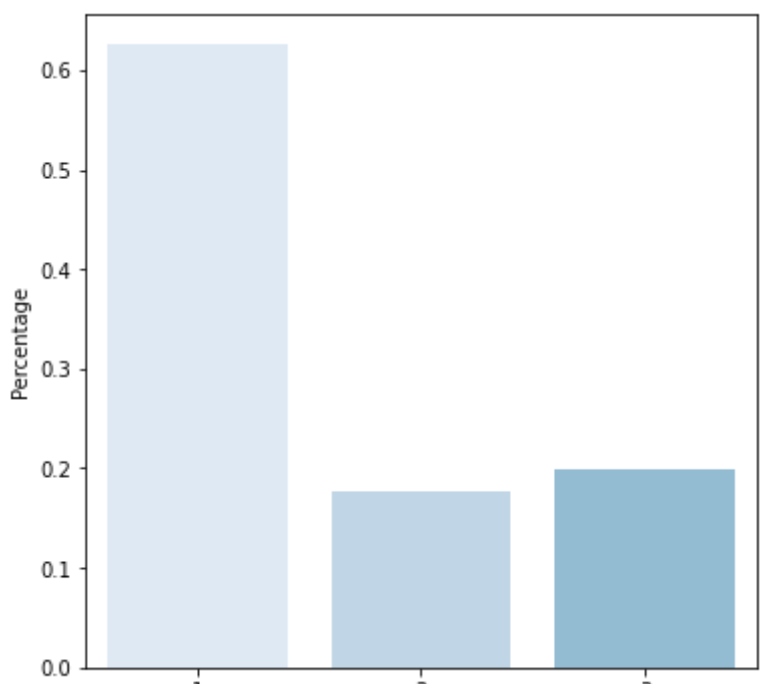


Question-3: Explore the data relations with target variable *origin*.

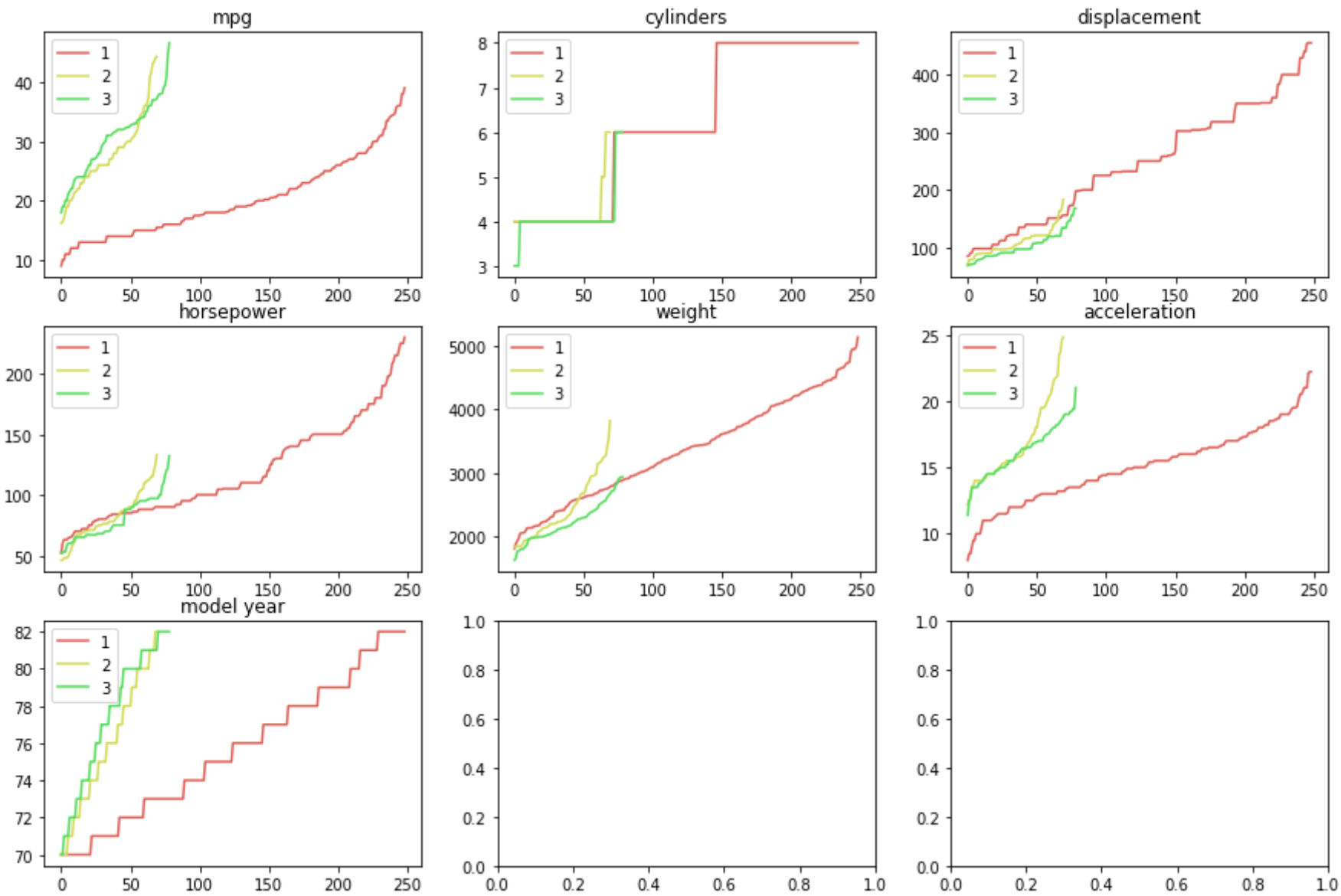
In [ ]:
```
Target_ratio=y.value_counts()/len(y)
print(Target_ratio)
```

```
1    0.625628
3    0.198492
2    0.175879
Name: origin, dtype: float64
```

In [ ]:
```
## Checking for data unbalance
plt.figure(figsize = (6,6))
sns.set_palette(sns.color_palette("Blues"))
sns.barplot(Target_ratio.index,Target_ratio,)
plt.ylabel('Percentage')
plt.show()
```



In [ ]:
```
sns.set_palette(sns.color_palette('hls'))
fig,ax=plt.subplots(3,3,figsize=(15,10))
col=data.columns
q=0
for i in ax:
    for j in i:
        for grp,tata in data.groupby('origin'):
            j.set_title(col[q])
            j.plot(sorted(tata[col[q]]))
            j.legend(np.arange(3)+1)
        if q==6:
            break
        q+=1
```



From the above data we can see that origin 1 has high stats for every feature while origin 3 and 2 have pretty low stats.

Question-4: Apply KNN model with k=3,9,12.

In [ ]:
```
X_train, X_test, y_train, y_test=train_test_split(X,y,random_state=0, test_size=0.3)
```

In [ ]:
```
knn9=KNeighborsClassifier(n_neighbors=9)
knn12=KNeighborsClassifier(n_neighbors=12)
knn3=KNeighborsClassifier(n_neighbors=3)
knn3.fit(X_train,y_train)
knn12.fit(X_train,y_train)
knn9.fit(X_train,y_train)
knn3_pred=knn3.predict(X_test)
knn12_pred=knn12.predict(X_test)
knn9_pred=knn9.predict(X_test)
```

Question-5: Evaluate the models and find the accuracy.

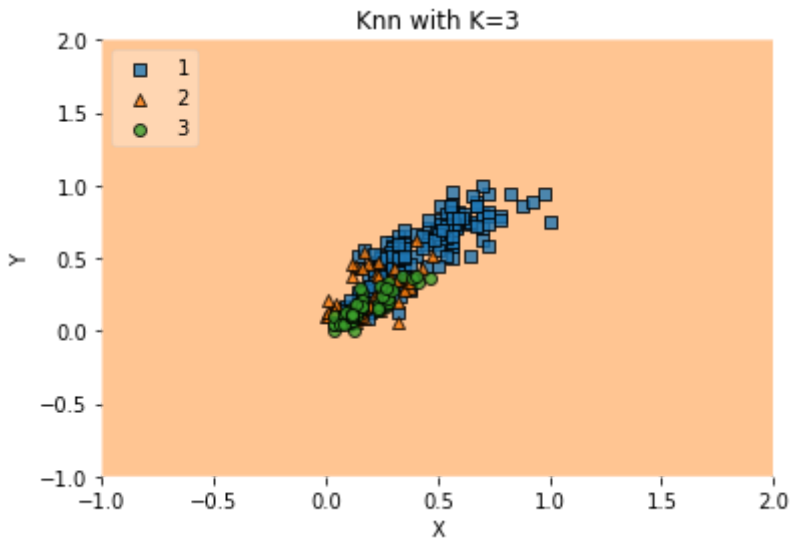
In [ ]:
```
print('Accuracy score when k=3:',accuracy_score(y_test,knn3_pred)*100)
print('Accuracy score when k=9:',accuracy_score(y_test,knn9_pred)*100)
print('Accuracy score when k=12:',accuracy_score(y_test,knn12_pred)*100)
```

```
Accuracy score when k=3: 70.0
Accuracy score when k=9: 77.5
Accuracy score when k=12: 71.66666666666667
```

Question-6: Plot the visualizations for the models.

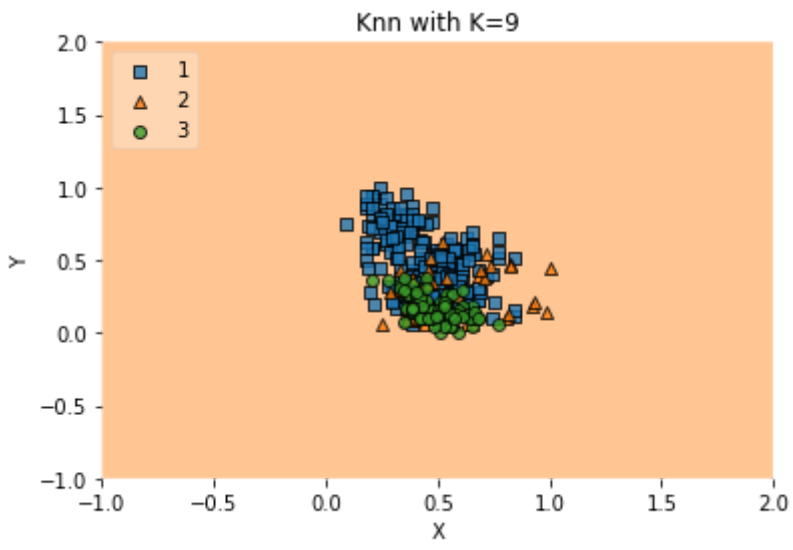
In [ ]:
```
value=4
width=4
plot_decision_regions(np.array(X), np.array(y), clf=knn3, legend=2,feature_index=[3,4],
              filler_feature_values={6: value, 1:value,2:value,5:value,0:value},
              filler_feature_ranges={6: width, 1: width,2: width,5: width,0: width})
# Adding axes annotations
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Knn with K='+ str(3))
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/mlxtend/plotting/decision_regions.py:242: UserWarning: No contour levels were
found within the data range.
  antialiased=True)
/usr/local/lib/python3.6/dist-packages/mlxtend/plotting/decision_regions.py:244: MatplotlibDeprecationWarning: Passin
g unsupported keyword arguments to axis() will raise a TypeError in 3.3.
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```



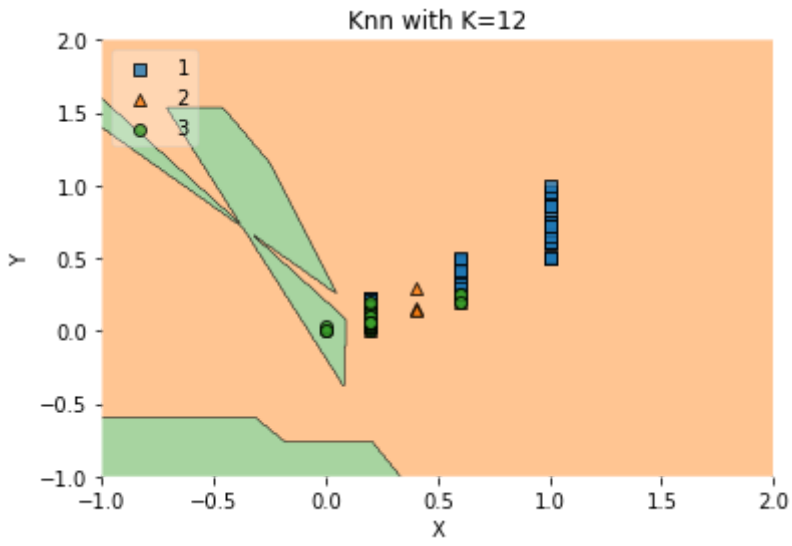
In [ ]:
```
value=4
width=4
plot_decision_regions(np.array(X), np.array(y), clf=knn9, legend=2,feature_index=[5,4],
              filler_feature_values={6: value, 1:value,2:value,3:value,0:value},
              filler_feature_ranges={6: width, 1: width,2: width,3: width,0: width})
# Adding axes annotations
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Knn with K='+ str(9))
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/mlxtend/plotting/decision_regions.py:242: UserWarning: No contour levels were
found within the data range.
  antialiased=True)
/usr/local/lib/python3.6/dist-packages/mlxtend/plotting/decision_regions.py:244: MatplotlibDeprecationWarning: Passin
g unsupported keyword arguments to axis() will raise a TypeError in 3.3.
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```



In [ ]:
```
value=4
width=4
plot_decision_regions(np.array(X), np.array(y), clf=knn12, legend=2,feature_index=[1,2],
                      filler_feature_values={6: value, 3:value,4:value,5:value,0:value},
                      filler_feature_ranges={6: width, 3: width,4: width,5: width,0: width})
# Adding axes annotations
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Knn with K='+ str(12))
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/mlxtend/plotting/decision_regions.py:244: MatplotlibDeprecationWarning: Passin
g unsupported keyword arguments to axis() will raise a TypeError in 3.3.
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```



"plot_decision_regions" can only be used to plot atmost 2 features only.

Scenario-3: Ergo Plastics

Ergo plastics were found in 1973, one of the oldest and largest plastic manufacturing industry across the world. They have been developing in a new plastic material which can withstand high pressure and temperature. They have created 2 new types of materials, and based on their elasticity and ductility; the data has been collected. The data has been collected based on material strength.

Problem Statement:

They have collected the data in a csv format. Task is to classify a given piece of plastic based on it's strength.

Dataset Description:

Attributes:

- **elasticity**: real
- **ductility**: real

Target Variable:

- **type**: tensil, brittle

Tasks to be Performed:

In order to attain the above goal below, tasks must be performed:

- Read the dataset and process all the missing values. - **Beginner**
- Split the data in training and testing set; Then apply support vector classifier. - **Beginner** (Bridging Question)
- Find the accuracy of the model and plot the confusion matrix. - **Intermediate**
- Plot the visualization of the model to check the boundaries. - **Advanced**

Topics Covered:

- Support Vector Machine

Question-1: Read the dataset and process all the missing values.

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
data=pd.read_csv('plastics_type.csv')
```

In [ ]:
```
data.head()
```

Out[ ]:

	elasticity	ductility	type
0	29.19393	8.443208	tensil
1	28.04907	7.237035	tensil
2	26.90421	7.719504	tensil
3	26.33178	7.478270	tensil
4	28.62150	8.684442	tensil

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 155 entries, 0 to 154
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   elasticity  154 non-null    float64
 1   ductility   152 non-null    float64
 2   type        155 non-null    object 
dtypes: float64(2), object(1)
memory usage: 3.8+ KB
```

In [ ]:
```
data.describe(include='all')
```

Out[ ]:

	elasticity	ductility	type
count	154.000000	152.000000	155
unique	NaN	NaN	2
top	NaN	NaN	brittle
freq	NaN	NaN	79
mean	33.427861	7.357653	NaN
std	4.702495	1.049866	NaN
min	24.614490	4.824690	NaN
25%	29.193930	6.754566	NaN
50%	33.200940	7.237035	NaN
75%	36.635520	7.960739	NaN
max	45.221970	10.614318	NaN

- Checking and processin Null Values

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
1	ductility	True	3	0.019355
0	elasticity	True	1	0.006452
2	type	False	0	0.000000

In [ ]:
```
print('Total Missing values: %s'%sum(miss.Count_))
```

```
Total Missing values: 4
```

In [ ]:
```
data.ductility.fillna(data.ductility.mean(),inplace=True)
data.elasticity.fillna(data.elasticity.mean(),inplace=True)
```

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in data.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	elasticity	False	0	0.0
1	ductility	False	0	0.0
2	type	False	0	0.0

Question-2: Split the data in training and testing set; Then apply support vector classifier.

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
```

In [ ]:
```
lb=LabelEncoder()
data.type=lb.fit_transform(data.type)
```

In [ ]:
```
X=data.drop('type',axis=1)
y=data.type
X_train, X_test, y_train, y_test=train_test_split(X,y,random_state=0, test_size=0.3)
```

In [ ]:
```
model=SVC(kernel='rbf')
model.fit(X_train,y_train)
```

Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
y_pred=model.predict(X_test)
```

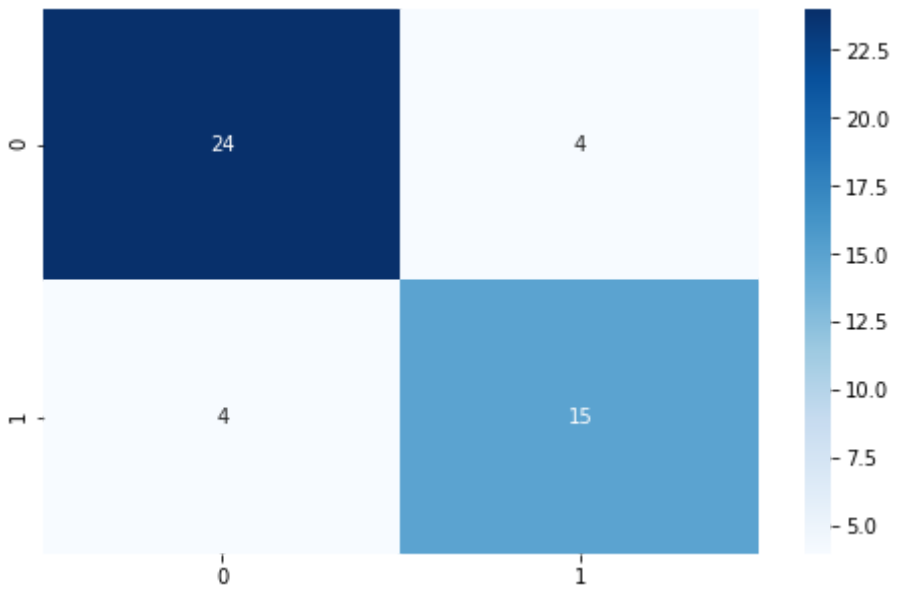
Question-3: Find the accuracy of the model and plot the confusion matrix.

In [ ]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from matplotlib.colors import ListedColormap
```

In [ ]:
```
print('The accuracy of the model is: ',round(accuracy_score(y_test,y_pred)*100,2))
```

```
The accuracy of the model is:  82.98
```

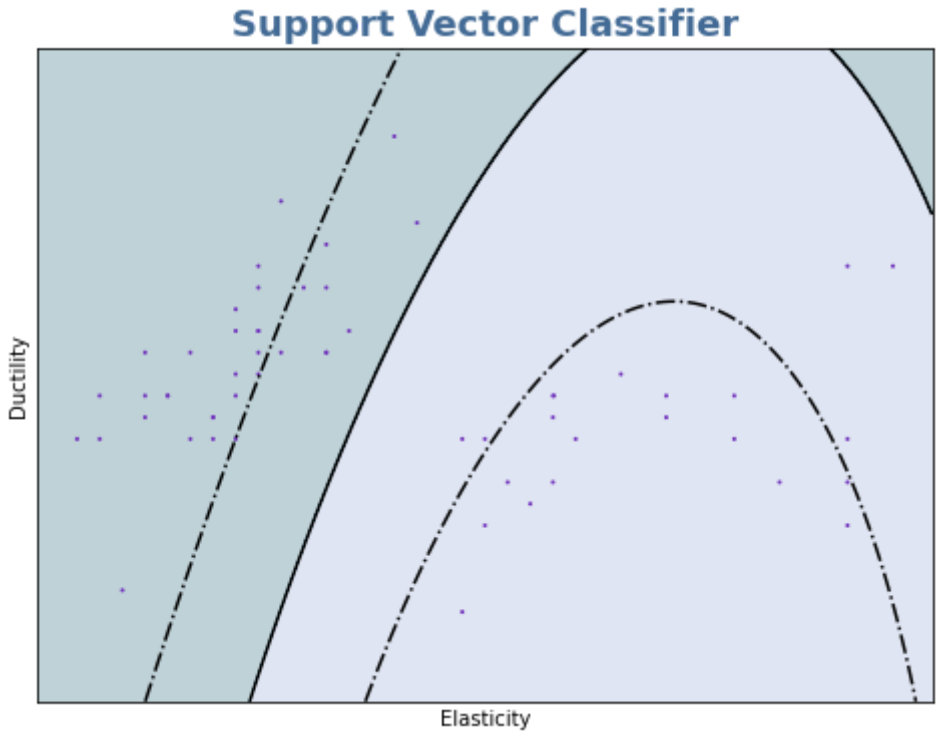
In [ ]:
```
lcm=confusion_matrix(y_test,y_pred)
lconf_matrix=pd.DataFrame(data=lcm)
plt.figure(figsize = (8,5))
sns.heatmap(lconf_matrix, annot=True,fmt='d',cmap='Blues')
plt.show()
```



Question-4: Plot the visualization of the model to check the boundaries.

In [ ]:
```
def plot(X,y,model):
    p=['tensil','brittle']
    X=np.array(X)
    X1, X2 = np.meshgrid(np.arange(start = X[:, 0].min() - 1, stop = X[:, 0].max() + 1,   step = 0.01),
                         np.arange(start = X[:, 1].min() - 1, stop = X[:, 1].max() + 1, step = 0.01))

    #To plot boundaries
    #In general, the space is divided into decision boundaries
    plt.figure(figsize=(8,6))
    Z=model.decision_function(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape)
    plt.contourf(X1, X2, model.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
                 alpha = 0.25, cmap = ListedColormap(('#809bce', '#004e64')))
    plt.contour(X1, X2, Z, colors=['k', 'k', 'k'],linestyles=['-.', '-', '-.'],levels=[-.7,0,.7])
    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())
    X=X.ravel()
    clr=[]
    for i in y:
        if i==0:
            clr.append('#ff499e')
        else:
            clr.append('#6f2dbd')
    plt.scatter(X[::2], X[1::2],y,c=clr, cmap= ListedColormap(('#ffaabb', '#23807A')))
    fm={'size':18,'color':'#436B95','weight':'bold'}
    plt.title('Support Vector Classifier',**fm)
    plt.xlabel('Elasticity')
    plt.ylabel('Ductility')
    plt.xticks([])
    plt.yticks([])
    return plt
```

In [ ]:
```
plot(X_train,y_train,model)
plt.show()
```



Module 1: Dimensionality Reduction

Principal Component Analysis (PCA)

Principal Component Analysis(PCA) is a dimensionality reduction technique in which we extract a new set of variables from the dataset. It is one of the widely used unsupervised algorithms. This technique can also be used for visualization, noise filtering, feature extraction or engineering, and much more.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique used to reduce the number of dimensions or features or variables in a dataset while retaining as much information as possible.

LDA is a **supervised** learning algorithm which is used for classification. Here, in LDA, we are interested in **maximizing the separability** betweeen the two groups to make the best decision. LDA is like PCA but focuses on maximizing the separability among known categories.

Scenario-1: U.S Nutrient Database

The dataset, U.S. Nutrient Database, includes the nutrient data from the national survey What We Eat In America, National Health and Nutrition Review Survey (WWEIA, NHANES), for determining dietary intakes. U.S. Nutrient data have historically been used for national nutrition monitoring

Earlier databases were composed mainly of commodity-type items such as wheat flour, sugar, milk, etc. However, with increased consumption of commercial processed and restaurant foods and changes in how national nutrition monitoring data are used, many commercial processed and restaurant items have been added to the database.

Problem Statement:

The increase of commercial commodities and fancy foods may create an imbalance in the nutrition of Americans. Hence the Department of Agriculture decides to maintain balance by finding out the right nutrition from the dataset. As the features are more, they choose to hire an Analyst who can help them by applying some of the Machine Learning techniques such as PCA and Kmeans clustering to reduce the components and visually analyze them.

Tasks to be performed:

To help out the Department of Agriculture below tasks should be performed,

- Import and preprocess the data- Beginner
- Fit and transform the data on the PCA model- Beginner
- Calculate the variance ratio and plot scree plot to find the principal components- Intermediate
- Fit and transform the PCA model for the principal components found- Beginner
- Create a Kmeans model for the PCA components- Intermediate
- Add the components and the clusters found by PCA and Kmeans to the scaled data- Advance
- Visualize the components based on its kmeans cluster- Intermediate
- Find out the rich nutrients in each component- Advance

Dataset Description:

The Dataset consist of 8618 rows and 45 columns, some of the major column descriptions are given below,

- **FoodGroup**- Category the food products belongs to
- **ShortDescrip**- Brief Description about the food group
- **CommonName**- Common name of the food
- **ScientificName**- Scientific name of the food
- **Energy_kcal**- Energy produced by food in calories
- **Protein_g**- Protein intake in grams
- **Fat_g**- Fat intake in grams
- **Carb_g**- Carbohydrates intake in grams
- **Sugar_g**- Sugar intake in grams
- **Fiber_g**- Fiber intake in grams
- **VitA_mcg VitB6_mg VitB12_mcg VitC_mg VitE_mg**- Different vitamins intake in milligrams
- **Calcium_mg**- Calcium intake in milligrmas
- **Iron_mg**- Iron intake in milligrams

Topics Covered:

- PCA
- Kmeans

In [1]:

```
# importing the libraries
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

In [2]:

```
!wget https://www.dropbox.com/s/s1n56r5siezy9zs/USA_Nutrition_Data.csv?dl=0
```

```
--2020-07-20 04:43:56--  https://www.dropbox.com/s/s1n56r5siezy9zs/USA_Nut
rition_Data.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:603
2:1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... conne
cted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/s1n56r5siezy9zs/USA_Nutrition_Data.csv [following]
--2020-07-20 04:43:56--  https://www.dropbox.com/s/raw/s1n56r5siezy9zs/USA
_Nutrition_Data.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc462a55549fb817661e3c7691e5.dl.dropboxusercontent.com/c
d/0/inline/A706v4ZDiIpR84RHfO-MfAeBR2NFfWr3e5ujctb5ZDHtJHuUc8H3Q0p6ZHIoq6l
lJvyzWxQ5KAKwzc4xH54q1X9pIVHaHVC02JMb_sQmq0vTcXToWnJ6PK7xGcQ7Dxlfei8/file#
[following]
--2020-07-20 04:43:56--  https://uc462a55549fb817661e3c7691e5.dl.dropboxus
ercontent.com/cd/0/inline/A706v4ZDiIpR84RHfO-MfAeBR2NFfWr3e5ujctb5ZDHtJHuU
c8H3Q0p6ZHIoq6llJvyzWxQ5KAKwzc4xH54q1X9pIVHaHVC02JMb_sQmq0vTcXToWnJ6PK7xGc
Q7Dxlfei8/file
Resolving uc462a55549fb817661e3c7691e5.dl.dropboxusercontent.com (uc462a55
549fb817661e3c7691e5.dl.dropboxusercontent.com)... 162.125.82.15, 2620:10
0:6032:15::a27d:520f
Connecting to uc462a55549fb817661e3c7691e5.dl.dropboxusercontent.com (uc46
2a55549fb817661e3c7691e5.dl.dropboxusercontent.com)|162.125.82.15|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 3065736 (2.9M) [text/plain]
Saving to: ‘USA_Nutrition_Data.csv?dl=0.1’

USA_Nutrition_Data. 100%[===================>]   2.92M  --.-KB/s    in 0.1
s

2020-07-20 04:43:57 (26.3 MB/s) - ‘USA_Nutrition_Data.csv?dl=0.1’ saved [3
065736/3065736]
```

Question-1: Import and pre-process the data (Beginner)

In [3]:

```
nutrients=pd.read_csv('/content/USA_Nutrition_Data.csv?dl=0')
nutrients.head()
```

Out[3]:

	ID	FoodGroup	ShortDescrip	Descrip	CommonName	MfgName	Scientific
0	1001	Dairy and Egg Products	BUTTER,WITH SALT	Butter, salted	NaN	NaN	
1	1002	Dairy and Egg Products	BUTTER,WHIPPED,WITH SALT	Butter, whipped, with salt	NaN	NaN	
2	1003	Dairy and Egg Products	BUTTER OIL,ANHYDROUS	Butter oil, anhydrous	NaN	NaN	
3	1004	Dairy and Egg Products	CHEESE,BLUE	Cheese, blue	NaN	NaN	
4	1005	Dairy and Egg Products	CHEESE,BRICK	Cheese, brick	NaN	NaN	

In [4]:

```
nutrients.shape
```

Out[4]:

```
(8618, 45)
```

In [5]:

```
nutrients.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8618 entries, 0 to 8617
Data columns (total 45 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ID                8618 non-null   int64  
 1   FoodGroup         8618 non-null   object 
 2   ShortDescrip      8618 non-null   object 
 3   Descrip           8618 non-null   object 
 4   CommonName        1063 non-null   object 
 5   MfgName           1560 non-null   object 
 6   ScientificName    732 non-null    object 
 7   Energy_kcal       8618 non-null   int64  
 8   Protein_g         8618 non-null   float64
 9   Fat_g             8618 non-null   float64
 10  Carb_g            8618 non-null   float64
 11  Sugar_g           8618 non-null   float64
 12  Fiber_g           8618 non-null   float64
 13  VitA_mcg          8618 non-null   int64  
 14  VitB6_mg          8618 non-null   float64
 15  VitB12_mcg        8618 non-null   float64
 16  VitC_mg           8618 non-null   float64
 17  VitE_mg           8618 non-null   float64
 18  Folate_mcg        8618 non-null   int64  
 19  Niacin_mg         8618 non-null   float64
 20  Riboflavin_mg     8618 non-null   float64
 21  Thiamin_mg        8618 non-null   float64
 22  Calcium_mg        8618 non-null   int64  
 23  Copper_mcg        8618 non-null   float64
 24  Iron_mg           8618 non-null   float64
 25  Magnesium_mg      8618 non-null   int64  
 26  Manganese_mg      8618 non-null   float64
 27  Phosphorus_mg     8618 non-null   int64  
 28  Selenium_mcg      8618 non-null   float64
 29  Zinc_mg           8618 non-null   float64
 30  VitA_USRDA        8618 non-null   float64
 31  VitB6_USRDA       8618 non-null   float64
 32  VitB12_USRDA      8618 non-null   float64
 33  VitC_USRDA        8618 non-null   float64
 34  VitE_USRDA        8618 non-null   float64
 35  Folate_USRDA      8618 non-null   float64
 36  Niacin_USRDA      8618 non-null   float64
 37  Riboflavin_USRDA  8618 non-null   float64
 38  Thiamin_USRDA     8618 non-null   float64
 39  Calcium_USRDA     8618 non-null   float64
 40  Copper_USRDA      8618 non-null   float64
 41  Magnesium_USRDA   8618 non-null   float64
 42  Phosphorus_USRDA  8618 non-null   float64
 43  Selenium_USRDA    8618 non-null   float64
 44  Zinc_USRDA        8618 non-null   float64
dtypes: float64(32), int64(7), object(6)
memory usage: 3.0+ MB
```

In [6]:

```
used = []
corrs = []
# The enumerate() function assigns an index to each item in an iterable object that can
be used to reference the item later
for i, j in enumerate(nutrients.corr().columns):
    for k in range(len(nutrients.corr())):
        if ((nutrients.corr().iloc[k, i] > 0.9) &
            (j not in used) &
            (j != nutrients.corr().index[k])):

            used.append(j)
            corrs.append((j, nutrients.corr().index[k],
                          np.round(nutrients.corr().iloc[k, i], 2)))


corrs_nutrients = pd.DataFrame([[i[0] for i in corrs],
                        [i[1] for i in corrs],
                        [i[2] for i in corrs]])

corrs_nutrients = corrs_nutrients.T.rename(columns = {0:'column',1:'row',2:'corr'})
corrs_nutrients[:15]
```

Out[6]:

	column	row	corr
0	VitA_mcg	VitA_USRDA	1
1	VitB6_mg	VitB6_USRDA	1
2	VitB12_mcg	VitB12_USRDA	1
3	VitC_mg	VitC_USRDA	1
4	VitE_mg	VitE_USRDA	1
5	Folate_mcg	Folate_USRDA	1
6	Niacin_mg	Niacin_USRDA	1
7	Riboflavin_mg	Riboflavin_USRDA	1
8	Thiamin_mg	Thiamin_USRDA	1
9	Calcium_mg	Calcium_USRDA	1
10	Copper_mcg	Copper_USRDA	1
11	Magnesium_mg	Magnesium_USRDA	1
12	Phosphorus_mg	Phosphorus_USRDA	1
13	Selenium_mcg	Selenium_USRDA	1
14	Zinc_mg	Zinc_USRDA	1

We can observe that the "_USRDA" features are similar to other nutrient features, and can be dropped

In [7]:

```
# dropping _USRDA features
nutrients.drop(nutrients.columns[nutrients.columns.str.contains('_USRDA')].values,
        inplace=True, axis=1)
```

Dropping first six categorical features as PCA and kmeans clustering does not affect them much

In [8]:

```
# dropping the categorical features as well
nutrients.set_index('ID', inplace=True)
nutrients_desc = nutrients.iloc[:, :6]
nutrients.drop(nutrients.columns[:6].values, axis=1, inplace=True)
```

We need to perform Scaling before applying PCA since PCA creates the Principal Component 1 in the direction of the maximum variance. But, if we do not scale, some features in our dataset might show high variance because of their larger values.

This is why it is strongly advisable to scale the data before applying PCA technique.

StandardScaler() will normalize each column of the dataset INDIVIDUALLY, so that each column or feature or variable will have mean = 0 and standard deviation = 1.

In [9]:

```
# standardizing the data
nutrients_TF = StandardScaler().fit_transform(nutrients)
```

In [10]:

```
# printing the shape of the data
nutrients_TF.shape
```

Out[10]:

```
(8618, 23)
```

Question-2: Fit and transform the data on the PCA model (Beginner) (Bridging Question)

In [11]:

```
# from sklearn.decomposition import PCA
pca = PCA()
pca.fit_transform(nutrients_TF)
```

Out[11]:

```
array([[-1.12177585e+00, -1.18225141e+00, -3.66193973e+00, ...,
         2.08470136e-01, -3.98100905e-02,  3.70138174e-02],
       [-1.11468691e+00, -1.18417302e+00, -3.66232928e+00, ...,
         2.22450074e-01, -3.76157809e-02,  3.68193054e-02],
       [-9.94919411e-01, -1.57357953e+00, -4.69772411e+00, ...,
         2.38667900e-01, -7.22173197e-02,  4.31958958e-02],
       ...,
       [-7.67670698e-01, -3.26765632e+00,  9.85205562e-01, ...,
         3.64559090e-01, -1.48468056e-01,  8.65889021e-02],
       [ 3.55897094e-01,  6.78435359e-01, -1.00293556e+00, ...,
        -8.24537105e-01,  1.12523136e-01, -4.86965696e-02],
       [-8.66889802e-01,  1.19845904e+00,  1.93486895e-01, ...,
        -1.53381540e-01,  2.30499506e-01, -1.91554534e-03]])
```

Question-3: Calculate the variance ratio and plot scree plot to find the principal components (Intermediate) (Bridging Question)

In [12]:

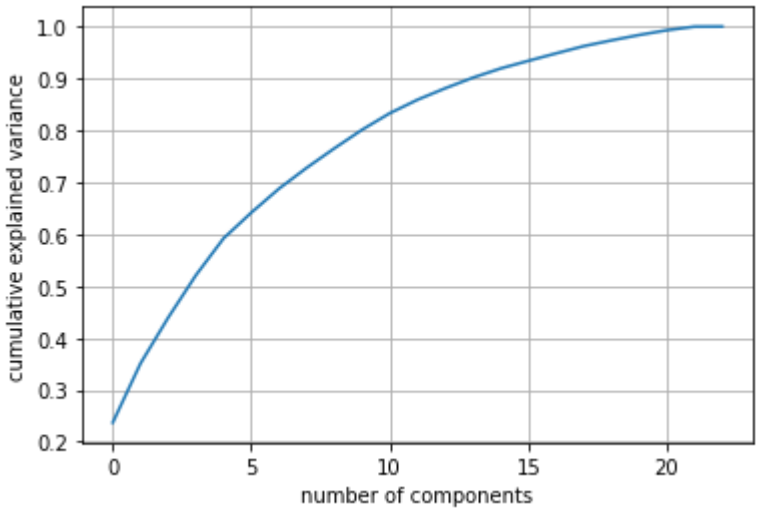
```
pca.explained_variance_ratio_
```

Out[12]:

```
array([2.36925468e-01, 1.13846015e-01, 8.83433734e-02, 8.17013669e-02,
       7.11161480e-02, 4.95813332e-02, 4.61246638e-02, 4.02719041e-02,
       3.74790508e-02, 3.58559322e-02, 3.17927198e-02, 2.59519549e-02,
       2.21153503e-02, 2.04113766e-02, 1.77385355e-02, 1.46927827e-02,
       1.43290735e-02, 1.39668837e-02, 1.11392349e-02, 1.03538827e-02,
       9.18082227e-03, 6.91711649e-03, 1.65010749e-04])
```

In [13]:

```
pca = PCA().fit(nutrients_TF)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.grid(True)
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



From above, we can observe that the explained variance ratio after 10 is negligible

Question-4: Fit and transform the PCA model for the principal components found (Beginner)

In [14]:

```
pca=PCA(n_components=10)
pca.fit(nutrients_TF)
```

Out[14]:

```
PCA(copy=True, iterated_power='auto', n_components=10, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [15]:

```
scores_pca=pca.transform(nutrients_TF)
```

In [16]:

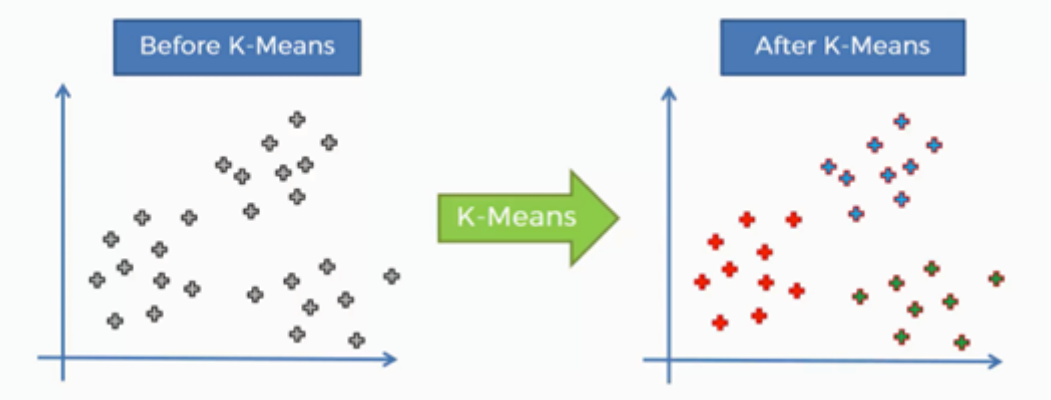
```
print(pca.explained_variance_ratio_[:10].sum())
```

```
0.8012450130873401
```

We can observe that first 5 eigenvectors account for almost 60% of the variance

K-Means

K-Means algorithm is an iterative algorithm that tries to partition the dataset into k pre-defined distinct non-overlapping clusters, where each data points belong to only one group



This figure implies how data is separated to particular categories using K-Means

Question-5: Create a Kmeans model for the PCA components (Intermediate)

In [17]:

```
#Set a 10 KMeans clustering
from sklearn.cluster import KMeans
kmeans_pca=KMeans(n_clusters=10, init='k-means++', random_state=42)
kmeans_pca.fit(nutrients_TF)
```

Out[17]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=10, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=42, tol=0.0001, verbose=0)
```

Question-6: Add the components and the clusters found by PCA and Kmeans to the scaled data (Advanced)

In [18]:

```
nutrients_seg_pca_kmeans=pd.concat([nutrients.reset_index(drop=True),pd.DataFrame(scores_pca)],axis=1)
nutrients_seg_pca_kmeans.columns.values[-10: ]=['Component1','Component2','Component3','Component4','Component5','Component6','Component7','Component8','Component9','Component10']
nutrients_seg_pca_kmeans['segment_kmeans_pca']=kmeans_pca.labels_
nutrients_seg_pca_kmeans.head()
```

Out[18]:

	Energy_kcal	Protein_g	Fat_g	Carb_g	Sugar_g	Fiber_g	VitA_mcg	VitB6_mg	VitB12_m
0	717	0.85	81.11	0.06	0.06	0.0	684	0.003	0
1	717	0.85	81.11	0.06	0.06	0.0	684	0.003	0
2	876	0.28	99.48	0.00	0.00	0.0	840	0.001	0
3	353	21.40	28.74	2.34	0.50	0.0	198	0.166	1
4	371	23.24	29.68	2.79	0.51	0.0	292	0.065	1

Question-7: Visualize the components (1 and 2) based on its kmeans cluster (Intermediate)

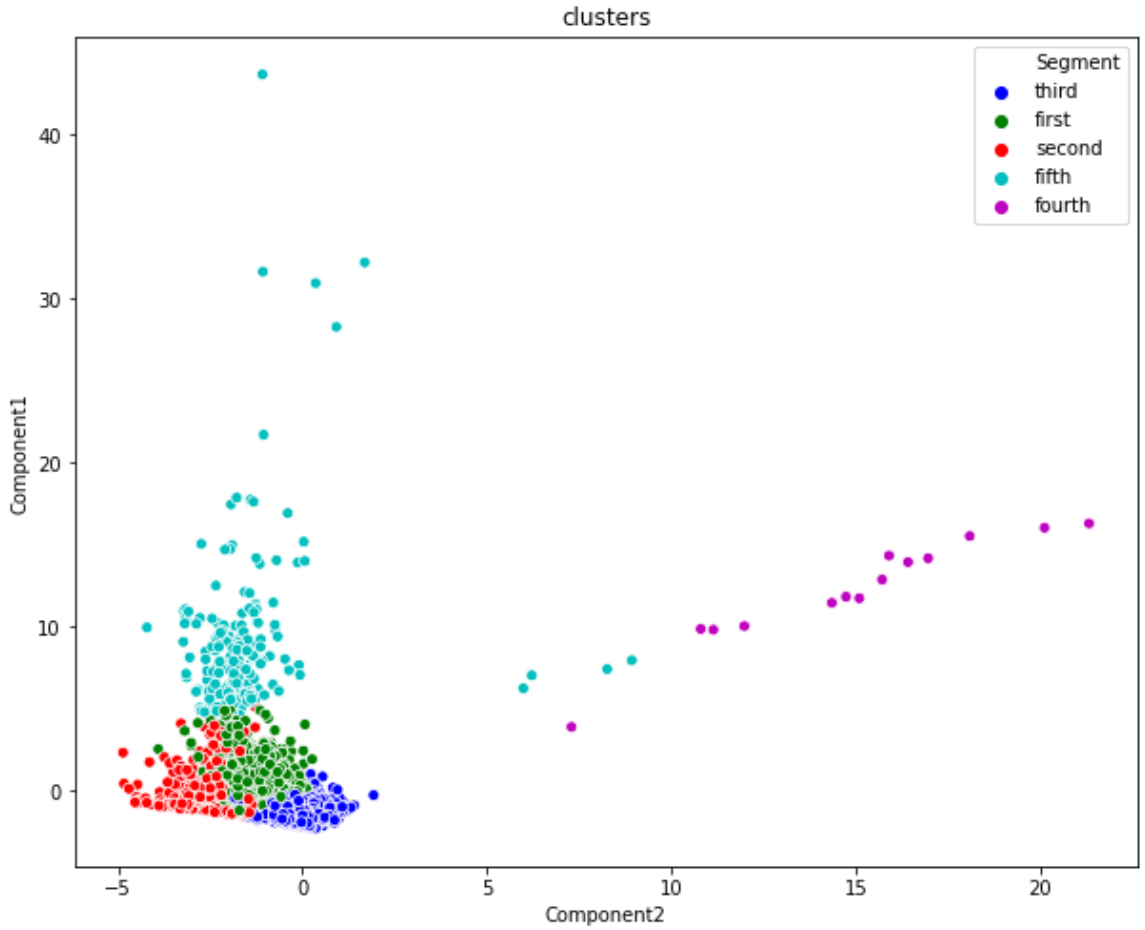
In [19]:

```
nutrients_seg_pca_kmeans['Segment']=nutrients_seg_pca_kmeans['segment_kmeans_pca'].map({0:'first',
                                                                                        1:'second',
                                                                                        2:'third',
                                                                                        3:'fourth',
                                                                                        4:'fifth'})
```

In [20]:

```
import seaborn as sns
x_axis= nutrients_seg_pca_kmeans['Component2']
y_axis= nutrients_seg_pca_kmeans['Component1']
plt.figure(figsize=(10,8))
sns.scatterplot(x_axis,y_axis,hue=nutrients_seg_pca_kmeans['Segment'],palette=['b','g',
'r','c','m'])
plt.title('clusters')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: F
utureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```



As we can see we have five different clusters based on components 1 and 2.

But we can further analyze for rest of the components by sorting them to find what nutrients each component is rich in.

Question-8: Find out which nutrient(s) is each component rich in(Advanced)

In [21]:

```
vects = pca.components_[:10]

first = pd.Series(vects[0], index=nutrients.columns)
first.sort_values(ascending=False)
```

Out[21]:

```
Riboflavin_mg    0.341325
Niacin_mg        0.337779
VitB6_mg         0.315663
Iron_mg          0.299857
Folate_mcg       0.284102
Thiamin_mg       0.272453
Zinc_mg          0.243551
Magnesium_mg     0.241348
Phosphorus_mg    0.199403
Fiber_g          0.181570
Copper_mcg       0.180806
VitB12_mcg       0.177985
Carb_g           0.169685
Calcium_mg       0.168112
Energy_kcal      0.157814
Protein_g        0.140620
VitE_mg          0.137122
VitA_mcg         0.133519
Manganese_mg     0.093567
Selenium_mcg     0.092319
VitC_mg          0.087639
Sugar_g          0.076323
Fat_g            0.033008
dtype: float64
```

First component is rich in Riboflavin_mg, Niacin_mg and VitB6

In [22]:

```
second = pd.Series(vects[1], index=nutrients.columns)
second.sort_values(ascending=False)
```

Out[22]:

```
VitB12_mcg       0.355045
Protein_g        0.343397
Selenium_mcg     0.239322
VitA_mcg         0.236470
Copper_mcg       0.212669
Zinc_mg          0.177798
Manganese_mg     0.088783
Phosphorus_mg    0.087447
Niacin_mg        0.084800
Riboflavin_mg    0.073471
VitB6_mg         0.021129
VitC_mg         -0.038525
Thiamin_mg      -0.075150
Iron_mg         -0.093812
Folate_mcg      -0.097093
Magnesium_mg    -0.103361
Calcium_mg      -0.105172
VitE_mg         -0.106372
Fat_g           -0.111670
Fiber_g         -0.257733
Energy_kcal     -0.273449
Sugar_g         -0.358769
Carb_g          -0.443416
dtype: float64
```

Second component is rich in VitaminB12 and Protein

In [23]:

```
third = pd.Series(vects[2], index=nutrients.columns)
third.sort_values(ascending=False)
```

Out[23]:

```
Folate_mcg       0.230985
Riboflavin_mg    0.192104
Thiamin_mg       0.184343
VitB6_mg         0.174642
Niacin_mg        0.164896
VitC_mg          0.162300
Iron_mg          0.087111
Sugar_g          0.055253
Carb_g           0.049814
VitB12_mcg      -0.012772
VitA_mcg        -0.021926
Zinc_mg         -0.038637
Fiber_g         -0.040392
Manganese_mg    -0.072629
Calcium_mg      -0.128145
Copper_mcg      -0.152259
Selenium_mcg    -0.163360
Magnesium_mg    -0.201230
VitE_mg         -0.207332
Protein_g       -0.213574
Phosphorus_mg   -0.274805
Energy_kcal     -0.462010
Fat_g           -0.534047
dtype: float64
```

Third component is rich in Folate, Riboflavin and Thiamin

In [24]:

```
fourth = pd.Series(vects[3], index=nutrients.columns)
fourth.sort_values(ascending=False)
```

Out[24]:

```
VitA_mcg         0.530395
Copper_mcg       0.389929
VitB12_mcg       0.346550
Manganese_mg     0.311369
Sugar_g          0.217372
Carb_g           0.174109
Energy_kcal      0.052281
VitC_mg          0.047584
Riboflavin_mg    0.046998
Fiber_g          0.042129
Fat_g            0.026520
VitE_mg          0.026174
Folate_mcg      -0.032482
Iron_mg         -0.059443
Magnesium_mg    -0.071804
Calcium_mg      -0.099340
Thiamin_mg      -0.103517
VitB6_mg        -0.114370
Niacin_mg       -0.156398
Selenium_mcg    -0.161623
Zinc_mg         -0.166322
Phosphorus_mg   -0.207871
Protein_g       -0.311111
dtype: float64
```

Fourth component is rich in Vitamin-A, copper and Vitamin-B12

In [25]:

```
fifth = pd.Series(vects[4], index=nutrients.columns)
fifth.sort_values(ascending=False)
```

Out[25]:

```
Calcium_mg       0.388702
Magnesium_mg     0.352749
Phosphorus_mg    0.344917
Fiber_g          0.332155
Copper_mcg       0.161236
Manganese_mg     0.125599
Iron_mg          0.097107
Carb_g           0.082731
VitC_mg          0.024716
Protein_g        0.013183
VitA_mcg         0.008133
Selenium_mcg    -0.005051
Sugar_g         -0.048427
VitB12_mcg      -0.058839
Zinc_mg         -0.064321
VitB6_mg        -0.133744
Folate_mcg      -0.137696
Riboflavin_mg   -0.153474
Thiamin_mg      -0.161509
Niacin_mg       -0.203439
VitE_mg         -0.238018
Energy_kcal     -0.293592
Fat_g           -0.394455
dtype: float64
```

Fifth Component is rich in Calcium, Magnesium and Phosphorus

In [26]:

```
sixth = pd.Series(vects[5], index=nutrients.columns)
sixth.sort_values(ascending=False)
```

Out[26]:

```
VitC_mg          0.545229
VitE_mg          0.475305
VitB6_mg         0.200545
Manganese_mg     0.182524
Magnesium_mg     0.140144
Fiber_g          0.134846
Fat_g            0.116477
Calcium_mg       0.070563
VitA_mcg         0.039010
Niacin_mg        0.026236
Iron_mg         -0.030548
Riboflavin_mg   -0.050791
Zinc_mg         -0.054547
Folate_mcg      -0.054685
Phosphorus_mg   -0.083833
Copper_mcg      -0.086231
VitB12_mcg      -0.123092
Thiamin_mg      -0.129962
Energy_kcal     -0.132849
Protein_g       -0.141441
Selenium_mcg    -0.243779
Carb_g          -0.279338
Sugar_g         -0.337048
dtype: float64
```

Sixth component is rich in VitC_mg, VitE_mg, VitB6_mg

In [27]:

```
seventh = pd.Series(vects[6], index=nutrients.columns)
seventh.sort_values(ascending=False)
```

Out[27]:

```
Calcium_mg       0.462875
VitC_mg          0.451293
Phosphorus_mg    0.323452
Sugar_g          0.298572
Riboflavin_mg    0.141714
VitA_mcg         0.101316
Energy_kcal      0.066848
VitB12_mcg       0.056180
Niacin_mg        0.045668
Fat_g            0.045295
VitB6_mg         0.024705
Carb_g           0.022128
Protein_g        0.006338
Selenium_mcg    -0.021905
Thiamin_mg      -0.071783
Zinc_mg         -0.105941
Copper_mcg      -0.119242
VitE_mg         -0.125431
Folate_mcg      -0.130930
Iron_mg         -0.133338
Manganese_mg    -0.136516
Magnesium_mg    -0.269565
Fiber_g         -0.414770
dtype: float64
```

Seventh component is rich in Calcium_mg, VitC_mg and Phosphorus_mg

In [28]:

```
eighth = pd.Series(vects[7], index=nutrients.columns)
eighth.sort_values(ascending=False)
```

Out[28]:

```
VitC_mg          0.516035
Selenium_mcg     0.410318
Magnesium_mg     0.191751
Copper_mcg       0.187807
Fiber_g          0.187637
Carb_g           0.155311
Sugar_g          0.143320
Protein_g        0.116493
VitB6_mg         0.083342
Energy_kcal      0.074527
Zinc_mg          0.033873
Niacin_mg        0.010346
VitB12_mcg       0.004982
Riboflavin_mg   -0.045033
VitA_mcg        -0.046358
Fat_g           -0.049692
Iron_mg         -0.107668
VitE_mg         -0.148535
Folate_mcg      -0.162167
Phosphorus_mg   -0.170576
Thiamin_mg      -0.190184
Manganese_mg    -0.345999
Calcium_mg      -0.366751
dtype: float64
```

Eighth component is rich in VitC_mg, Selenium_mcg and Magnesium_mg

In [29]:

```
ninth = pd.Series(vects[8], index=nutrients.columns)
ninth.sort_values(ascending=False)
```

Out[29]:

```
Manganese_mg     0.757045
Selenium_mcg     0.358340
Sugar_g          0.150992
Protein_g        0.148609
Niacin_mg        0.120639
Carb_g           0.117971
VitC_mg          0.085600
Energy_kcal      0.041631
Magnesium_mg     0.031621
Thiamin_mg       0.026895
VitB6_mg         0.023107
Phosphorus_mg   -0.012741
VitA_mcg        -0.022078
Riboflavin_mg   -0.045001
VitE_mg         -0.061840
Folate_mcg      -0.064154
Fiber_g         -0.065872
Fat_g           -0.091669
Iron_mg         -0.108841
Zinc_mg         -0.108957
VitB12_mcg      -0.131888
Calcium_mg      -0.135789
Copper_mcg      -0.361794
dtype: float64
```

Ninth component is rich in Manganese_mg, Selenium_mcg and Sugar_g

In [30]:

```
tenth = pd.Series(vects[9], index=nutrients.columns)
tenth.sort_values(ascending=False)
```

Out[30]:

```
Thiamin_mg       0.438182
Fat_g            0.200809
Riboflavin_mg    0.195356
VitC_mg          0.155420
Energy_kcal      0.140973
VitA_mcg         0.118238
Magnesium_mg     0.110565
Niacin_mg        0.088607
Fiber_g          0.086742
Phosphorus_mg    0.061936
Copper_mcg       0.047175
Protein_g        0.018151
Folate_mcg       0.017563
Manganese_mg    -0.018196
Carb_g          -0.044210
Selenium_mcg    -0.075577
Calcium_mg      -0.080433
Iron_mg         -0.083149
VitB12_mcg      -0.133593
VitB6_mg        -0.198273
Sugar_g         -0.387842
VitE_mg         -0.424257
Zinc_mg         -0.477749
dtype: float64
```

Tenth component is rich in Thiamin_mg, Fat_g and Riboflavin_mg

Inference:

Now instead of going through all the features in the dataset. Nutritionist from the Department of Agriculture can go through these five components and create a new nutrition plan for Americans

Scenario-2: Velcro Winery

Velcro Winery is one of the finest Wineries in Italy. They prepare three different types of wine, which are represented using 178 samples. They have performed 13 different chemical analyses on each of their samples and recorded them in the form of a dataset.

Problem Statement:

The winery CEO is interested in classifying the wines into its respective classes based on its chemical analyses. Hence, he decides to hire you as an Analyst who can apply the Dimensionality reduction technique (LDA) to classify the wines and build a model to check its accuracy.

Tasks to be performed:

- Importing and preprocessing the data is a priority- Beginner
- Build an LDA model using sklearn- Beginner
- Visualize the LDA components with the help of model- Intermediate
- Split the data into Training and Testing. (keep the test as 30% and the random state as 2)- Beginner
- Build a Random forest classifier model (keep random state as 2)- Intermediate
- Calculate the accuracy of the model- Beginner
- Create a confusion matrix to verify the prediction-Intermediate
- Visualize the confusion matrix using heatmap- Advance

Dataset Description

The dataset contains 178 rows and 14 columns which is described as,

- **alcohol**- Alcohol content in wine
- **malic_acid**- Amount of malic acid present in wine
- **ash**- Amount of ash present in wine
- **alcalinity_of_ash**- Amount of alcalinity of ash present in wine
- **magnesium**- Amount of magnesium present in wine
- **total_phenols**- Amount of total phenols present in wine
- **flavanoids**- Amount of total flavored phenols present in wine
- **nonflavanoid_phenols**- Amount of total non-flavored phenols present in wine
- **proanthocyanins**- Amount of proanthocyanins present in wine
- **color_intensity**- color intensity of the wine
- **hue**- pH range of wine which inturn decides the acidity of the wine
- **od280/od315_of_diluted_wines**- Diluted wines.
- **proline**- Amino acid present in grapes or wines
- **class**- class to which wines belong

Topic Covered

LDA

Question-1: Import and pre-process the data (Beginner)

In [31]:

```
# importing the libraries
from sklearn.datasets import load_wine
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [32]:

```
w = load_wine()
x = pd.DataFrame(w.data, columns=w.feature_names)
y = pd.Categorical.from_codes(w.target, w.target_names)
x.head()
```

Out[32]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

In [33]:

```
#joining both train and the target variable
w1 = x.join(pd.Series(y, name='class'))
w1.head()
```

Out[33]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

Question-2: Build an LDA model using sklearn (Beginner)

In [34]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA()
X_lda = lda.fit_transform(x, y)
```

Question-3: Visulaize the LDA components with the help of model (Intermediate)

As matplotlib cannot handle categorical values, you must convert it into numerical values

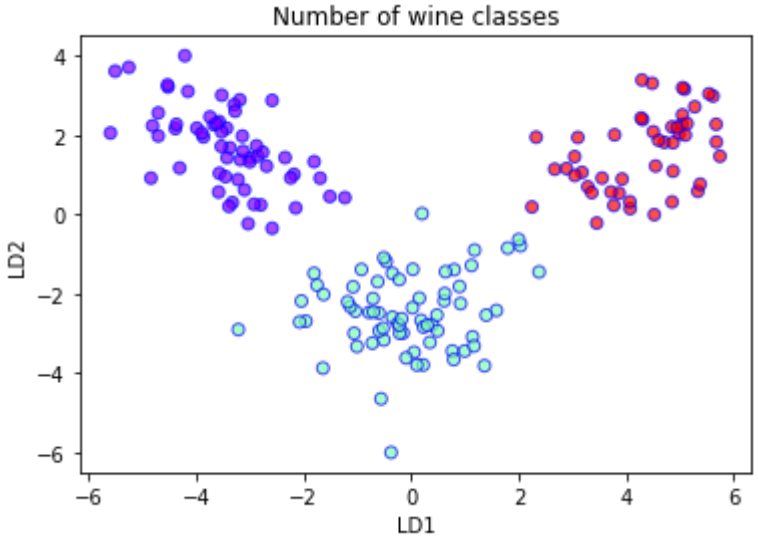
This can be done using label encoder

In [35]:

```
l_encoder = LabelEncoder()
y = l_encoder.fit_transform(w1['class']) #transforming the categorical value to numeric
al value
```

In [36]:

```
#classifying the classes using LDA
plt.scatter(X_lda[:,0],X_lda[:,1],c=y,cmap='rainbow',alpha=0.7,edgecolors='b')
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.title('Number of wine classes')
plt.show()
```



Question-4: Split the data into Training and Testing. (keep the test as 30% and random state as 2) (Beginner)

In [37]:

```
X_train, X_test, y_train, y_test = train_test_split(w1.drop(['class'],axis = 'columns'
),w.target,test_size = 0.3,random_state=2)
```

Question-5: Build a Randomforest classifier model (keep random state as 2) (Intermediate)

In [38]:

```
model = RandomForestClassifier(random_state=0)
model.fit(X_train,y_train)
```

Out[38]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbo
se=0,
                       warm_start=False)
```

In [39]:

```
y_predicted = model.predict(X_test)
```

Question-6: Calculate the accuracy of the model (Beginner)

In [40]:

```
print('Accuracy of the model is: ' + str(accuracy_score(y_test, y_predicted)))
```

```
Accuracy of the model is: 0.9629629629629629
```

Question-7: Create a confusion matrix to verify the prediction (Intermediate)

In [41]:

```
c_matrix = confusion_matrix(y_test,y_predicted)
c_matrix
```

Out[41]:

```
array([[18,  2,  0],
       [ 0, 20,  0],
       [ 0,  0, 14]])
```

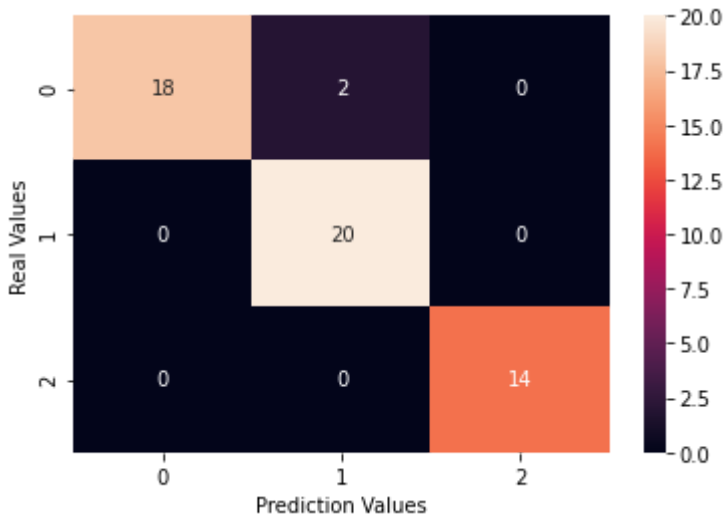
Question-8: Visualize the confusion matrix using heatmap (Advanced)

In [42]:

```
sns.heatmap(c_matrix,annot = True)
plt.xlabel('Prediction Values')
plt.ylabel('Real Values')
```

Out[42]:

```
Text(33.0, 0.5, 'Real Values')
```



We can infer that class 1 and class 2 are prefectly predicted where in class 3 predicted one wrong value

Question-9: Create a classification report using sklearn

In [43]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predicted, digits=3))
```

```
              precision    recall  f1-score   support

           0      1.000     0.900     0.947        20
           1      0.909     1.000     0.952        20
           2      1.000     1.000     1.000        14

    accuracy                          0.963        54
   macro avg      0.970     0.967     0.967        54
weighted avg      0.966     0.963     0.963        54
```

Scenario-3: FIFA

FIFA Dataset is a football simulation video game developed by EA Vancouver as part of Electronic Arts' FIFA series.It has plenty of attributes covering all aspects of a real-life footballer in an attempt to imitate him as much as possible in the virtual world.

Problem Statement:

As EA Vancouver CEO decides to add a premier league in his next update of the game. He wishes to know how players are targeted through Transfer Window (A gap after a season), he wants to analyze the players through their attributes rather than their reputation to make the game as real as possible.

You being an Analyst, are expected to perform PCA to reduce the dimensions along with kmeans to find out the hidden patterns in the data.

Tasks to be performed:

To attain the above objective below, tasks should be performed:

- Importing and preprocessing the data is priority- Beginner
- Identify the appropriate number of components by building a PCA model and scree plot- Intermediate
- Fit and transform the PCA model again for the principal components found- Intermediate

Dataset Description:

The dataset contains 89 columns and 18207 rows of FIFA Datasets by EA Vancouver. Here's a brief description of the 10 columns in the dataset:

- **ID** - Id of Each player
- **Name** - Name of each player
- **Age** - Age of each player
- **Nationality** - Nationality of each player
- **Overall** - Overall performance of each player
- **Potential** - Potential of each player
- **Club** - Name of the Club each player belong to
- **Value** - Net value of each player
- **Wage** - Amount each player earns per match
- **Preferred Foot** - Preferred foot of the player's (Right or Left)

Topics Covered:

- PCA

Question-1: Importing and preprocessing the data is priority (Beginner)

In [44]:

```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [45]:

```
!wget https://www.dropbox.com/s/2m892ugv2m8mmnu/Fifa.csv?dl=0
```

```
--2020-07-20 04:45:32--  https://www.dropbox.com/s/2m892ugv2m8mmnu/Fifa.cs
v?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:603
2:1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... conne
cted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/2m892ugv2m8mmnu/Fifa.csv [following]
--2020-07-20 04:45:32--  https://www.dropbox.com/s/raw/2m892ugv2m8mmnu/Fif
a.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc0a8976e4aced3ec061cf1ab6df.dl.dropboxusercontent.com/c
d/0/inline/A71TNt09vs9QcETTmttD_XGVbylSfmLIjrJJXebSKTrLVlYR-x4XIEbW1p2QF4Y
TdjAmPifzoUFdJ1VorhTmRe1OE0JAQ3ikGWUg5Mp_3WFqU8wjWrfe1yBuGeEKgZ160zU/file#
[following]
--2020-07-20 04:45:32--  https://uc0a8976e4aced3ec061cf1ab6df.dl.dropboxus
ercontent.com/cd/0/inline/A71TNt09vs9QcETTmttD_XGVbylSfmLIjrJJXebSKTrLVlYR
-x4XIEbW1p2QF4YTdjAmPifzoUFdJ1VorhTmRe1OE0JAQ3ikGWUg5Mp_3WFqU8wjWrfe1yBuGe
EKgZ160zU/file
Resolving uc0a8976e4aced3ec061cf1ab6df.dl.dropboxusercontent.com (uc0a8976
e4aced3ec061cf1ab6df.dl.dropboxusercontent.com)... 162.125.82.15, 2620:10
0:6032:15::a27d:520f
Connecting to uc0a8976e4aced3ec061cf1ab6df.dl.dropboxusercontent.com (uc0a
8976e4aced3ec061cf1ab6df.dl.dropboxusercontent.com)|162.125.82.15|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 9140113 (8.7M) [text/plain]
Saving to: ‘Fifa.csv?dl=0.2’

Fifa.csv?dl=0.2     100%[===================>]   8.72M  18.9MB/s    in 0.5
s

2020-07-20 04:45:33 (18.9 MB/s) - ‘Fifa.csv?dl=0.2’ saved [9140113/914011
3]
```

In [46]:

```
fifa= pd.read_csv("/content/Fifa.csv?dl=0")
fifa.head()
```

Out[46]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	h
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	h
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	h
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	h
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	

5 rows × 89 columns

In [47]:

```
#Dropping the unnecessary columns
fifa.drop(columns=['Unnamed: 0'], inplace=True)
```

In [48]:

```
fifa.columns
```

Out[48]:

```
Index(['ID', 'Name', 'Age', 'Photo', 'Nationality', 'Flag', 'Overall',
       'Potential', 'Club', 'Club Logo', 'Value', 'Wage', 'Special',
       'Preferred Foot', 'International Reputation', 'Weak Foot',
       'Skill Moves', 'Work Rate', 'Body Type', 'Real Face', 'Position',
       'Jersey Number', 'Joined', 'Loaned From', 'Contract Valid Until',
       'Height', 'Weight', 'LS', 'ST', 'RS', 'LW', 'LF', 'CF', 'RF', 'RW',
       'LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM', 'RCM', 'RM', 'LWB', 'LDM',
       'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB', 'RCB', 'RB', 'Crossing',
       'Finishing', 'HeadingAccuracy', 'ShortPassing', 'Volleys', 'Dribbli
ng',
       'Curve', 'FKAccuracy', 'LongPassing', 'BallControl', 'Acceleratio
n',
       'SprintSpeed', 'Agility', 'Reactions', 'Balance', 'ShotPower',
       'Jumping', 'Stamina', 'Strength', 'LongShots', 'Aggression',
       'Interceptions', 'Positioning', 'Vision', 'Penalties', 'Composure',
       'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving', 'GKHandli
ng',
       'GKKicking', 'GKPositioning', 'GKReflexes', 'Release Clause'],
      dtype='object')
```

We are only interested in those attributes that constitute the skillset of a football player.

In [49]:

```
#creating a dataframe fifa_opa by selecting only the required columns
fifa_opa = fifa[['Name', 'Age', 'Overall', 'Potential', 'Value', 'International Reputat
ion', 'Height', 'Weight', 'Crossing', 'Wage','Club','Nationality','Finishing','HeadingA
ccuracy','ShortPassing','Volleys','Dribbling','Curve','FKAccuracy','LongPassing','BallC
ontrol','Acceleration','SprintSpeed','Agility','Reactions','Balance','ShotPower','Jumpi
ng','Stamina','Strength','LongShots','Aggression','Interceptions','Positioning','Visio
n','Penalties','Composure','Marking','StandingTackle','SlidingTackle']]
fifa_opa.head()
```

Out[49]:

	Name	Age	Overall	Potential	Value	International Reputation	Height	Weight	Crossing	Wage
0	L. Messi	31	94	94	€110.5M	5.0	5'7	159lbs	84.0	€565K
1	Cristiano Ronaldo	33	94	94	€77M	5.0	6'2	183lbs	84.0	€405K
2	Neymar Jr	26	92	93	€118.5M	5.0	5'9	150lbs	79.0	€290K
3	De Gea	27	91	93	€72M	4.0	6'4	168lbs	17.0	€260K
4	K. De Bruyne	27	91	92	€102M	4.0	5'11	154lbs	93.0	€355K

All the players in the dataset are sorted by their **'Overall'** value. We can reorder the samples in the dataset as 'Clustering' algorithms get biased by order

In [50]:

```
# Considering 100 samples from data
fifa_opa = fifa_opa.sample(100)
fifa_opa.head()
```

Out[50]:

	Name	Age	Overall	Potential	Value	International Reputation	Height	Weight	Crossing	Wa
6411	Héctor	27	69	70	€975K	1.0	5'7	150lbs	68.0	€
9757	José Lara	18	66	81	€1.3M	1.0	5'6	132lbs	66.0	€
14538	P. Klimala	19	61	73	€450K	1.0	6'0	168lbs	29.0	€
8662	F. Heerkens	28	67	67	€625K	1.0	6'0	176lbs	55.0	€
14776	E. Ntim	22	60	69	€270K	1.0	5'10	159lbs	56.0	€

We need to focus only on the attributes of a footballer like **Passing, Shooting, Dribbling etc** and identify those footballers who are similar

In [51]:

```
# Focussing only on the attributes
fifa_pep = fifa_opa.drop([ 'Name', 'Club','Nationality','Overall','Potential','Height',
'Weight','Wage','Value'], axis=1)
fifa_pep.head()
```

Out[51]:

	Age	International Reputation	Crossing	Finishing	HeadingAccuracy	ShortPassing	Volleys	Drib
6411	27	1.0	68.0	42.0	60.0	66.0	44.0	
9757	18	1.0	66.0	62.0	45.0	66.0	58.0	
14538	19	1.0	29.0	59.0	59.0	51.0	55.0	
8662	28	1.0	55.0	20.0	66.0	65.0	23.0	
14776	22	1.0	56.0	21.0	57.0	54.0	25.0	

We need to scale the features as some of the features like `Age` are in the range 16-40 whereas `Crossing` is in the range 0-100. We will use **MinMax scaling** since we want discover clusters with reference to the best and worst values of players across different attributes.

In [52]:

```
#scaling the data
from sklearn.preprocessing import MinMaxScaler
X_min = MinMaxScaler().fit_transform(fifa_pep)
X_min
```

Out[52]:

```
array([[0.55      , 0.        , 0.79452055, ..., 0.61904762, 0.7125    ,
        0.74358974],
       [0.1       , 0.        , 0.76712329, ..., 0.26190476, 0.325     ,
        0.23076923],
       [0.15      , 0.        , 0.26027397, ..., 0.39285714, 0.1       ,
        0.06410256],
       ...,
       [0.4       , 0.        , 0.47945205, ..., 0.51190476, 0.625     ,
        0.64102564],
       [0.6       , 0.        , 0.76712329, ..., 0.3452381 , 0.175     ,
        0.20512821],
       [0.65      , 0.        , 0.08219178, ..., 0.04761905, 0.075     ,
        0.1025641 ]])
```

Question-2: Identify the appropriate number of components by building a PCA model and scree plot (Intermediate)

Since we are dealing with a huge number of attributes - 31, our analysis will be paralyzed by the **'Curse of Dimensionality'**. So, let us perform **Principal Component Analysis** to identify those principal components that explain most of the variance in the dataset.

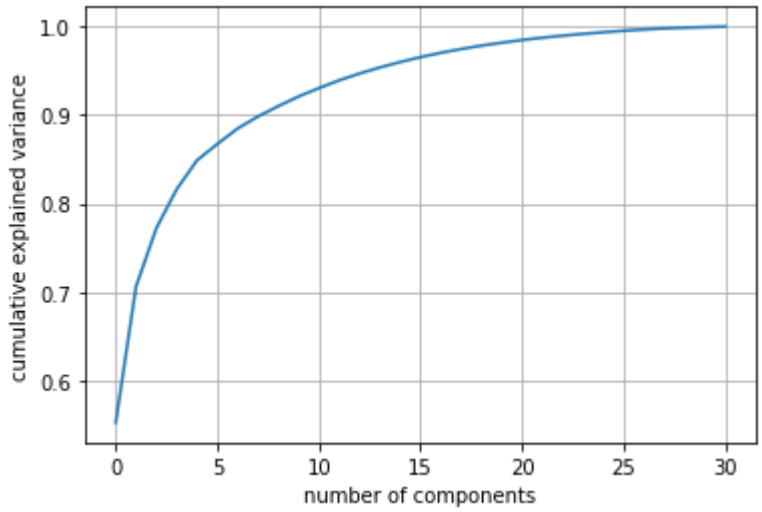
In [53]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=31)
pca.fit(X_min)
print(pca.explained_variance_ratio_)
```

```
[0.55275158 0.15379054 0.06581369 0.04397402 0.03261722 0.01844656
 0.01751212 0.01350337 0.01171921 0.01090983 0.00938798 0.00888674
 0.00762482 0.00685003 0.00617077 0.00546295 0.00493291 0.00425995
 0.00385875 0.00326864 0.00305522 0.00261683 0.00227944 0.00204828
 0.00181035 0.00169885 0.00140874 0.00126531 0.00078955 0.00066611
 0.00061964]
```

In [54]:

```
pca = PCA().fit(X_min)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.grid(True)
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



After applying PCA, we can see that the dimensionality has been reduced from 31 to just 10

Question-3: Fit and transform the PCA model again for the principal components found (Intermediate) (Bridging Question)

In [55]:

```
pca=PCA(n_components=10)
pca.fit(X_min)
scores_pca=pca.transform(X_min)
```

In [56]:

```
print(pca.explained_variance_ratio_[:10].sum())
```

```
0.9210381423805312
```

From the above scree plot, we can notice that 10 principal components are explaining 93% of the variance in the dataset

Question-4: Build a Kmeans model using the transformed vector and write your inference

In [57]:

```
#let's build clusters using the Transformed_vector
from sklearn.cluster import KMeans
num_of_clusters = range(2,25)
error = []

for num_clusters in num_of_clusters:
    clusters = KMeans(num_clusters)
    clusters.fit(scores_pca)
    error.append(clusters.inertia_/100)

temp=pd.DataFrame({"Cluster_Numbers":num_of_clusters, "Error_Term":error})
temp
```

Out[57]:

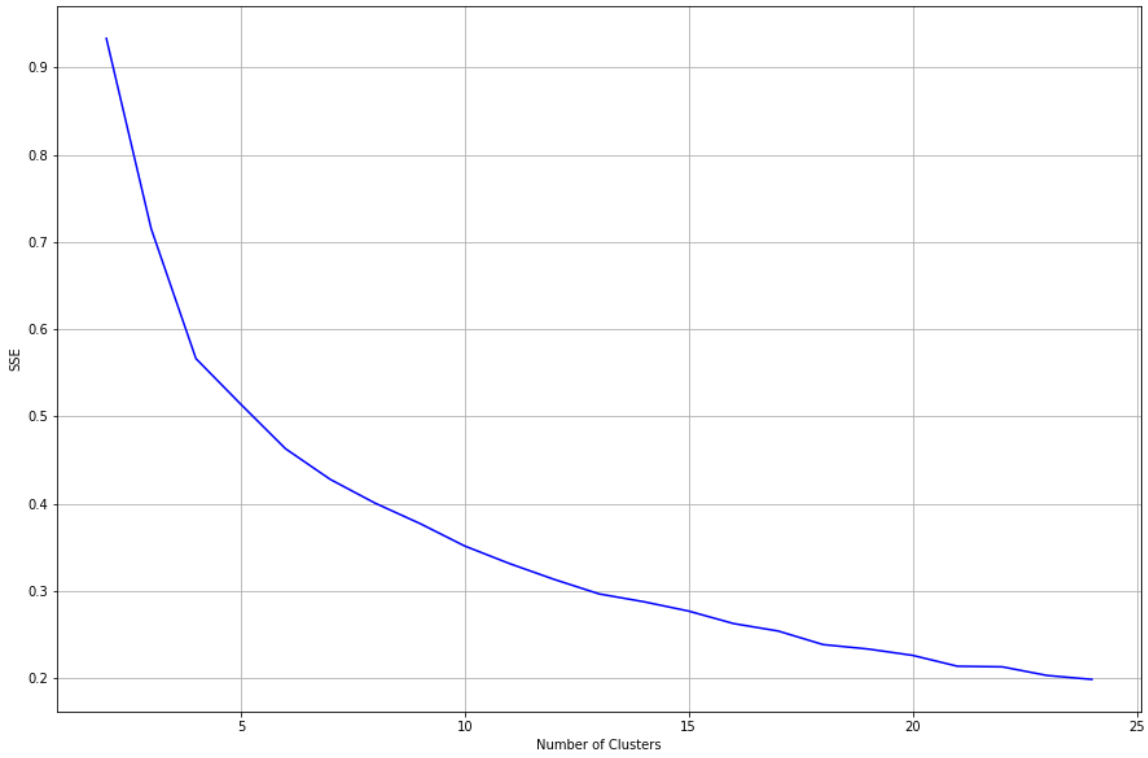
	Cluster_Numbers	Error_Term
0	2	0.933549
1	3	0.715438
2	4	0.566414
3	5	0.514169
4	6	0.463076
5	7	0.427928
6	8	0.400692
7	9	0.377376
8	10	0.351560
9	11	0.331431
10	12	0.313243
11	13	0.296642
12	14	0.287650
13	15	0.276956
14	16	0.262676
15	17	0.254073
16	18	0.238552
17	19	0.233500
18	20	0.226183
19	21	0.213735
20	22	0.213038
21	23	0.203158
22	24	0.198637

We can observe that there are so many error term hence, we can apply elbow method to find the right number of clusters

Question-5: Create a Kmeans model with the help Elbow method

In [58]:

```
#Find the right number of clusters
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(15,10))
plt.plot(temp.Cluster_Numbers, temp.Error_Term,  color='blue')
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
plt.grid(True)
plt.show()
```



From the above elbow plot, it seems convergence started happening when the no.of clusters is **'5'**

In [59]:

```
#Build 5 clusters
clusters1 = KMeans(5)
clusters1.fit(scores_pca)
clusters1.labels_
```

Out[59]:

```
array([4, 0, 0, 4, 3, 2, 4, 1, 0, 0, 0, 2, 0, 2, 2, 0, 0, 1, 1, 4, 4, 2,
       4, 4, 1, 4, 0, 3, 4, 2, 0, 3, 0, 0, 3, 2, 0, 4, 1, 3, 0, 4, 4, 4,
       2, 3, 2, 1, 4, 2, 4, 2, 4, 1, 0, 2, 4, 2, 3, 4, 3, 0, 1, 4, 4, 2,
       3, 1, 0, 2, 4, 0, 4, 3, 0, 2, 4, 4, 1, 3, 3, 0, 0, 2, 4, 1, 0, 3,
       0, 2, 0, 0, 3, 4, 3, 0, 1, 4, 0, 1], dtype=int32)
```

Question-6: Analyze the above model and find the pattern in the data (Infer the same)

We have to make some Changes,

we have to convert the string type currency to integer inorder to apply some matematical operations

In [60]:

```
#converting the string currency to integer.

def currencyconversion(amount):
    n_amount = []
    for s in amount:
        list(s)
        abbr = s[-1]
        if abbr is 'M':
            s = s[1:-1]
            s = float(''.join(s))
            s *= 1000000
        elif abbr is 'K':
            s = s[1:-1]
            s = float(''.join(s))
            s *= 1000
        else:
            s = 0
        n_amount.append(s)
    return n_amount
```

In [61]:

```
#Applying the above function to convert string to integer
fifa_opa['Value'] = currencyconversion(list(fifa_opa['Value']))
fifa_opa['Wage'] = currencyconversion(list(fifa_opa['Wage']))
```

In [62]:

```
# Apply some matematical operations which makes the analysis easier
players = fifa_opa['Name']
wage = fifa_opa['Wage']/1000
overall = fifa_opa['Overall']
age = fifa_opa['Age']
value = fifa_opa['Value']/100000
```

Create a Dataframe consisting of clusters, players, Wages in thousands, overall, age, and Transfer value in millions

In [63]:

```
dff = pd.DataFrame({'clusters': clusters1.labels_, 'players': players, 'Wages in thousa
nds':wage, 'overall':overall, 'age':age, 'Transfer value in millions':value})
dff.sort_values('clusters')
```

Out[63]:

	clusters	players	Wages in thousands	overall	age	Transfer value in millions
15174	0	M. Molina	2.0	60	18	4.25
10092	0	E. Hooi	3.0	65	26	5.75
8954	0	F. Varela	3.0	66	22	10.00
16974	0	K. Henry	1.0	56	18	1.50
6180	0	L. James	4.0	69	25	13.00
...	...	...	...	...	...	...
11526	4	D. Myrestam	1.0	64	31	3.00
1948	4	B. Pearson	12.0	75	23	80.00
6803	4	T. Kane	9.0	68	24	10.00
5555	4	A. Bernardini	3.0	70	31	12.00
6411	4	Héctor	5.0	69	27	9.75

100 rows × 6 columns

Group those clusters and describe them statistically

In [64]:

```
dff.groupby('clusters').describe().transpose()
```

Out[64]:

clusters		0	1	2	3	4
Wages in thousands	count	27.000000	13.000000	18.000000	15.000000	27.000000
	mean	5.074074	4.384615	24.944444	1.266667	9.740741
	std	7.054327	4.752867	24.426293	1.032796	20.504395
	min	1.000000	1.000000	2.000000	1.000000	0.000000
	25%	1.000000	1.000000	5.000000	1.000000	2.000000
	50%	2.000000	2.000000	19.500000	1.000000	3.000000
	75%	4.500000	6.000000	35.500000	1.000000	8.000000
	max	33.000000	15.000000	96.000000	5.000000	105.000000
overall	count	27.000000	13.000000	18.000000	15.000000	27.000000
	mean	66.037037	63.461538	72.888889	56.200000	66.703704
	std	4.926237	7.698651	5.245602	3.876670	5.856061
	min	56.000000	53.000000	63.000000	51.000000	59.000000
	25%	62.500000	58.000000	69.500000	54.000000	63.000000
	50%	66.000000	65.000000	72.500000	55.000000	65.000000
	75%	69.000000	66.000000	76.000000	58.500000	68.500000
	max	78.000000	78.000000	85.000000	65.000000	84.000000
age	count	27.000000	13.000000	18.000000	15.000000	27.000000
	mean	23.037037	24.538462	28.888889	20.600000	26.000000
	std	3.837883	5.562005	4.085684	4.256088	3.990373
	min	18.000000	18.000000	22.000000	16.000000	20.000000
	25%	20.000000	20.000000	25.250000	18.500000	23.000000
	50%	23.000000	22.000000	30.000000	19.000000	25.000000
	75%	26.000000	29.000000	32.500000	21.000000	30.000000
	max	30.000000	36.000000	35.000000	34.000000	33.000000
Transfer value in millions	count	27.000000	13.000000	18.000000	15.000000	27.000000
	mean	17.305556	14.023077	45.805556	1.853333	26.927778
	std	25.867482	28.337899	51.029155	1.461832	67.023081
	min	1.500000	0.700000	2.500000	0.200000	0.000000
	25%	4.625000	1.400000	10.500000	0.950000	3.250000
	50%	9.500000	2.800000	30.500000	1.400000	4.500000
	75%	13.000000	6.500000	62.500000	2.250000	9.875000
	max	130.000000	95.000000	180.000000	6.000000	340.000000

Inferences,

- If you are looking at controlling the **wages and transfer fee** and still get the brightest talent in the Dutch league, then you should focus on players in **Cluster 3**. The overall of players in this cluster is at par with those in other clusters and they have lower wages and transfer value than players in other clusters.
- If you have **unlimited budget to spend** and are focused only on getting the **top players** in the league, then you should focus on buying players in Clusters 0,1 and 4.
- You need to be a bit careful with players in **Cluster 2** as there seem to be a lot of **outliers having high wages**

Unsupervised Learning

K-Means Clustering

K-Means Clustering Algorithm is simple and easy to implement **unsupervised** machine learning algorithm. The main objective of K-means clustering is to group similar data points into a cluster. It is used when we have unlabelled data. This algorithm aims to divide 'n' number of observations into 'k' number of clusters.

A **cluster** refers to a collection of data points clubbed together because of existing similarities. The task of identifying and assigning similar data-points to a cluster is known as **Clustering**.

Hierarchical Clustering

Hierarchical Clustering is an unsupervised learning algorithm that is used to group similar data-points in a cluster. It creates clusters with a pre-determined order from top to bottom—for example, files and folders organized in a hierarchy on a hard-disk.

There are two types of Hierarchical Clustering:

- **Agglomerative Hierarchical Clustering** :

It is most commonly used. It works in a **bottom-up manner**. Here, we assign each data-point to an individual cluster and then calculate the similarity between the clusters using either **Euclidean Distance** or **Manhattan Distance** and club the most similar clusters. It merges similar points of the cluster and stops when all the data-points are combined into a single cluster.

- **Divisive Hierarchical Clustering** :

It is not used much. It is the inverse of Agglomerative Hierarchical Clustering. It works in a top-down manner. Here, we assign all the data points to a single cluster after each iteration. We remove the data-points from the cluster, which are not similar, and each data-point that we remove is treated as an individual cluster. Here, we are dividing the cluster in every step. This is why it is known as Divisive Hierarchical Clustering.

Scenario-1: Bigmart

Bigmart is a supermarket mall, with some primary data about customers like Customer ID, age, gender, annual income, and spending Score. Spending Score is something they assign to the customer based on their defined parameters like customer behavior and purchasing data.

Customer segmentation is one of the significant applications of K-means Clustering. It is the practice of dividing a customer base into groups of individuals having similar features. It is a useful method because Bigmart can approach these different customer segments and distribute marketing resources effectively.

Problem Statement:

Bigmart CEO wants to understand the customers who can converge easily [Target Customers] so that the marketing team can have a sense and plan the strategy accordingly

So, being an Analyst, you must

1. Achieve customer segmentation using the machine learning algorithm (KMeans Clustering) in Python
2. Identify target customers with whom you can start marketing strategy

Tasks to be performed:

To attain the above goal below, tasks must be performed:

Importing and preprocessing the data is priority

- Segregate the columns to be clustered (**Age** and **Spending Score (1-100)**)- Beginner
- Identify the value of K using Elbow curve- Beginner
- Create, fit and predict K_means model with the help of clusters found by Elbow method- Intermediate
- Calculate the centroids of each cluster- Intermediate
- Visualize customer based on their ***Age*** and ***Spending Score (1-100)***- Advanced

Dataset Description:

The Bigmart Dataset contains 200 rows and 5 columns

- **Customer Id**- Identity number of a Customer
- **Gender**- Gender of the Customer
- **Age**- Age of the Customer
- **Annual Income (k$)**- Annual income of the Customer in Dollars
- **Spending Score (1-100)**- Spending score of customer

Topics Covered:

- Elbow Method
- K-Means Clustering

Importing and preprocessing the Data

In [1]:

```
# importing the required library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
!wget https://www.dropbox.com/s/no4nw204xhep7ov/Bigmart.csv?dl=0
```

```
--2020-07-20 04:32:23--  https://www.dropbox.com/s/no4nw204xhep7ov/Bigmar
t.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:
1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connec
ted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/no4nw204xhep7ov/Bigmart.csv [following]
--2020-07-20 04:32:24--  https://www.dropbox.com/s/raw/no4nw204xhep7ov/Big
mart.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucd118f8ac12f8e594f5faebcd3f.dl.dropboxusercontent.com/c
d/0/inline/A726-XrGvw-U90vMtWnjm_lpD5pKGMgRhneYPE2GIoB35pi9SMO9gaWuPYZZuSu
J1_BI097mpYd5VKQzrmobBmh9mBwXRsJxHrIXsT8B5uMzRe5oYGPrO61TNSmb_NNiHs8/file#
[following]
--2020-07-20 04:32:24--  https://ucd118f8ac12f8e594f5faebcd3f.dl.dropboxus
ercontent.com/cd/0/inline/A726-XrGvw-U90vMtWnjm_lpD5pKGMgRhneYPE2GIoB35pi9
SMO9gaWuPYZZuSuJ1_BI097mpYd5VKQzrmobBmh9mBwXRsJxHrIXsT8B5uMzRe5oYGPrO61TNS
mb_NNiHs8/file
Resolving ucd118f8ac12f8e594f5faebcd3f.dl.dropboxusercontent.com (ucd118f8
ac12f8e594f5faebcd3f.dl.dropboxusercontent.com)... 162.125.5.15, 2620:100:
601d:15::a27d:50f
Connecting to ucd118f8ac12f8e594f5faebcd3f.dl.dropboxusercontent.com (ucd1
18f8ac12f8e594f5faebcd3f.dl.dropboxusercontent.com)|162.125.5.15|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 3981 (3.9K) [text/plain]
Saving to: ‘Bigmart.csv?dl=0’

Bigmart.csv?dl=0    100%[===================>]   3.89K  --.-KB/s    in 0s

2020-07-20 04:32:24 (491 MB/s) - ‘Bigmart.csv?dl=0’ saved [3981/3981]
```

In [3]:

```
customer=pd.read_csv('/content/Bigmart.csv?dl=0')
customer.head()
```

Out[3]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [4]:

```
# check the shape of the data
print(f'This dataset has {customer.shape[0]} rows and {customer.shape[1]} columns.')
```

```
This dataset has 200 rows and 5 columns.
```

In [5]:

```
# check for null data
customer.isnull().sum()
```

Out[5]:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Dataset doesnot have any null values

In [6]:

```
# Dataset info
customer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Question-1: Segregate the columns to be clustered (Age and Spending Score (1-100)) (Beginner)

In [7]:

```
#considering the columns we have to cluster
Y= customer.iloc[:, [2,4]].values
```

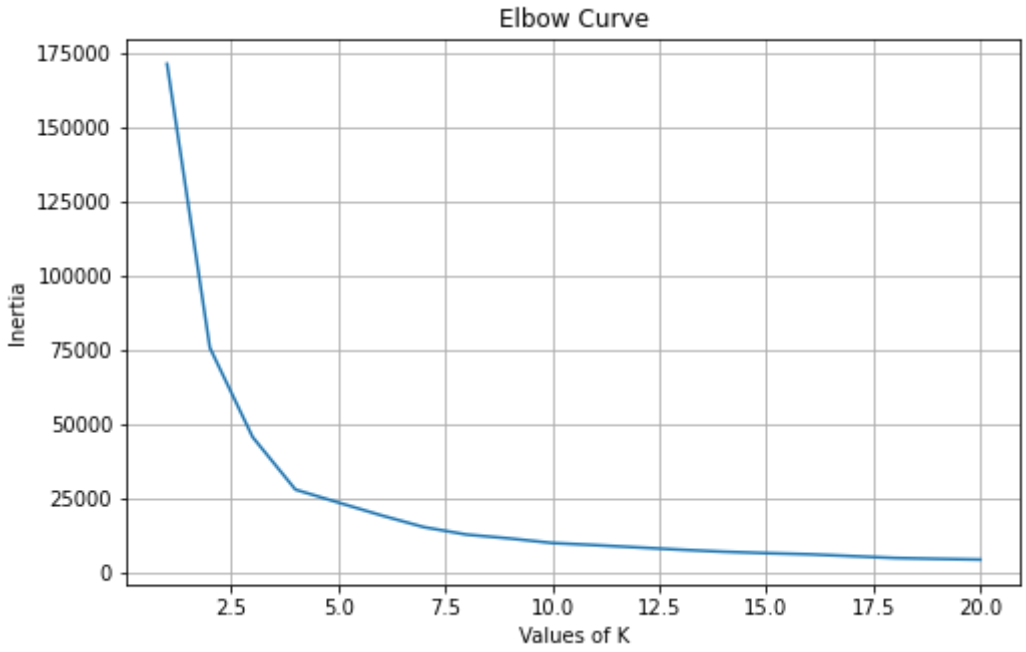
Question-2: Identify the value of K using Elbow curve (Beginner) (Bridging Question)

In [8]:

```
inertia_list = []
for num_clusters in np.arange(1,21):
    kmeans =KMeans(n_clusters=num_clusters)
    kmeans.fit(Y)
    inertia_list.append(kmeans.inertia_)
```

In [9]:

```
#Plotting the Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1, 21), inertia_list)
plt.grid(True)
plt.xlabel('Values of K')
plt.ylabel('Inertia')
plt.title('Elbow Curve')
plt.show()
```



From above, we select the optimum value of k by determining the Elbow Point - a point after which the inertia starts decreasing linearly. In this case, we can choose the value of k as 4

Question-3: Create, fit and predict K_means model with the help of clusters found by Elbow method (Intermediate) (Bridging Question)

In [10]:

```
kmeansmodel = KMeans(n_clusters= 4, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(Y)
```

K-means starts with allocating cluster centers randomly and then looks for "better" solutions. K-means++ starts with randomly allocating one cluster center and then searches for other centers given the first one.

Hence we use K-means++

Question-4: Calculate the centroids of each clusters (Intermediate) (Bridging Question)

A centroid is a data point (imaginary or real) at the center of a cluster.

In [11]:

```
#printing the centroids of each cluster
centroids = kmeans.cluster_centers_
centroids
```

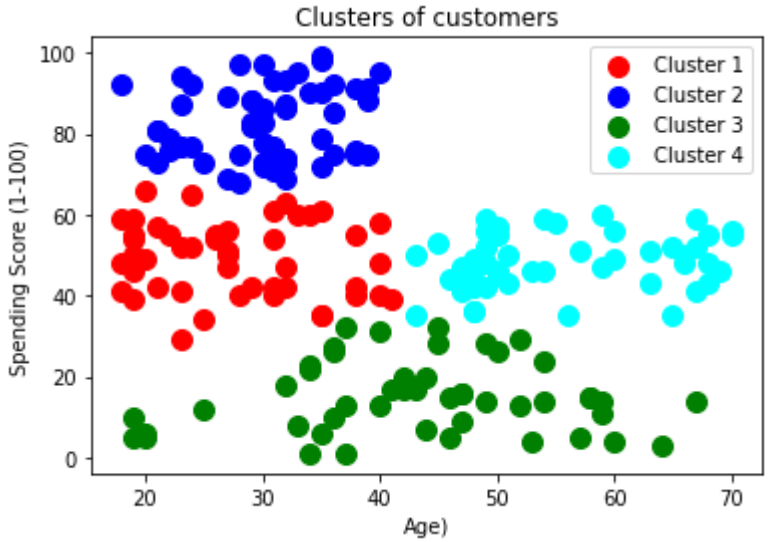
Out[11]:

```
array([[44.1       , 16.5       ],
       [33.11111111, 59.66666667],
       [65.        , 44.44444444],
       [23.        , 90.8       ],
       [34.4       , 23.2       ],
       [48.29411765, 45.58823529],
       [22.        , 53.05882353],
       [32.25      , 73.55      ],
       [34.64285714, 93.5       ],
       [47.5       ,  6.25      ],
       [20.6       ,  7.6       ],
       [51.7       , 56.        ],
       [58.8       , 10.8       ],
       [38.55555556, 36.55555556],
       [21.9       , 75.8       ],
       [66.22222222, 54.        ],
       [35.33333333,  6.5       ],
       [30.66666667, 85.33333333],
       [26.07692308, 40.46153846],
       [49.66666667, 30.44444444]])
```

Question-5: Visualize customer based on their *Age* and *Spending Score (1-100)* (Advanced)

In [12]:

```
plt.scatter(Y[y_kmeans == 0, 0], Y[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Clus
ter 1')
plt.scatter(Y[y_kmeans == 1, 0], Y[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Clu
ster 2')
plt.scatter(Y[y_kmeans == 2, 0], Y[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cl
uster 3')
plt.scatter(Y[y_kmeans == 3, 0], Y[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Clu
ster 4')
plt.title('Clusters of customers')
plt.xlabel('Age)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



We have 4 clusters of Customer based on Age and Spending Score (1-100):

- **Cluster-1 (Green)**: Customers of this group have a low score no matter the age. Perhaps the marketing team should develop a different strategy and a different approach to giving these custumers a new perspective which will increase their scores
- **Cluster-2 (Blue)**: Customers of this group are young and adults age < 40 with medium scores between 35 to 75. To increase this group's count, maybe the marketing team can create new actions for young public like games, customized products, and others
- **Cluster-3 (cyan)**: Customers of this group are 40 and above whose scores are medium. To increase the score of this group, the marketing team can come up with calm places inside the mall, like new restaurants, typical food courts, and others
- **Cluster-4 (Red)**: These customers are with highest spending score. The marketing team should pay close attention to retain these customers

Scenario-2: Caltech Bank

Caltech bank is based out of Sydney and plans a new loaning scheme for its customers and wants to analyze its customer data to determine how the customer's earning is associated with their credit score.

A credit score is a number ranging from 300-850, which shows the creditworthiness of a customer. That is the number of open accounts, total levels of debt, and repayment history. Higher the credit score, the more trustworthy is the customer.

Problem Statement:

Our objective here is to use the clustering method, to find the high credit score clusters of customers. It will summarize the existing loan scheme and help Caltech bank to decide on the new loan scheme

Tasks to be performed:

In order to attain the above goal below, tasks must be performed:

Importing and preprocessing the data is priority

- Segregate the columns to be clustered (**Earning** and **Credit Score**)- Beginner
- Identify the number of clusters using Dendrogram- Intermediate
- Apply a threshold value to the above visualization (dendrogram) to find the clusters- Intermediate
- Create, fit and predict hierarchical model with the help of clusters found by Dendrogram-Intermediate
- Visualize customer based on their ***Earning*** and ***Credit Score***- Advanced

Dataset Description:

The dataset consist of 4 columns and 200 rows,

- **Cust_id**- Id of the Customer
- **Age**- Age of the Customer
- **Earning**- Earning of the Customer
- **Credit Score**- Credit score of the Customer

Topics Covered:

- Dendrograms
- Hierarchical Clustering

Import and preprocess the Data

In [13]:

```
#importing the libtrary
import numpy as np # Linear Algebra
import pandas as pd # Data Processing
import matplotlib.pyplot as plt # Matplotlib for Plotting
import seaborn as sns
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
import warnings
warnings.filterwarnings('ignore')
```

In [14]:

```
!wget https://www.dropbox.com/s/uysjhtgf8mdj1tv/Caltech_Bank-Customers.csv?dl=0
```

```
--2020-07-20 04:32:30--  https://www.dropbox.com/s/uysjhtgf8mdj1tv/Calte
ch_Bank-Customers.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601
d:1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... conn
ected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/uysjhtgf8mdj1tv/Caltech_Bank-Customers.csv [following]
--2020-07-20 04:32:30--  https://www.dropbox.com/s/raw/uysjhtgf8mdj1tv/C
altech_Bank-Customers.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc8951f8dbc62b9c899df33d7a2d.dl.dropboxusercontent.co
m/cd/0/inline/A722bQk_efoI7sZXGVxK-klMJ2_AMiJyBYvzAv1CG5uugUAFHaqYAmdDM2
IfjFbPVoLmDZvkPP9u3f_6-4S-cVwFDqnbvZ3l6p0n4ZbW4j0dSm-mCXnE4E-u3KYBUgJXZJ
E/file# [following]
--2020-07-20 04:32:30--  https://uc8951f8dbc62b9c899df33d7a2d.dl.dropbox
usercontent.com/cd/0/inline/A722bQk_efoI7sZXGVxK-klMJ2_AMiJyBYvzAv1CG5uu
gUAFHaqYAmdDM2IfjFbPVoLmDZvkPP9u3f_6-4S-cVwFDqnbvZ3l6p0n4ZbW4j0dSm-mCXnE
4E-u3KYBUgJXZJE/file
Resolving uc8951f8dbc62b9c899df33d7a2d.dl.dropboxusercontent.com (uc8951
f8dbc62b9c899df33d7a2d.dl.dropboxusercontent.com)... 162.125.5.15, 2620:
100:601d:15::a27d:50f
Connecting to uc8951f8dbc62b9c899df33d7a2d.dl.dropboxusercontent.com (uc
8951f8dbc62b9c899df33d7a2d.dl.dropboxusercontent.com)|162.125.5.15|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2726 (2.7K) [text/plain]
Saving to: ‘Caltech_Bank-Customers.csv?dl=0’

Caltech_Bank-Custom 100%[===================>]   2.66K  --.-KB/s    in 0
s

2020-07-20 04:32:31 (328 MB/s) - ‘Caltech_Bank-Customers.csv?dl=0’ saved
[2726/2726]
```

In [15]:

```
loan=pd.read_csv('/content/Caltech_Bank-Customers.csv?dl=0')
loan.head()
```

Out[15]:

	Cust_id	Age	Earning	Credit Score
0	1	19	15	39
1	2	21	15	81
2	3	20	16	6
3	4	23	16	77
4	5	31	17	40

In [16]:

```
#check for null values
loan.isnull().sum()
```

Out[16]:

```
Cust_id         0
Age             0
Earning         0
Credit Score    0
dtype: int64
```

There is no null values in the dataset

In [17]:

```
# check the shape of the data
print(f'This dataset has {loan.shape[0]} rows and {loan.shape[1]} columns.')
```

```
This dataset has 200 rows and 4 columns.
```

In [18]:

```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Cust_id       200 non-null    int64
 1   Age           200 non-null    int64
 2   Earning       200 non-null    int64
 3   Credit Score  200 non-null    int64
dtypes: int64(4)
memory usage: 6.4 KB
```

Question-1: Segregate the columns to be clustered (Beginner)

(**Earning** and **Credit Score**)

In [19]:

```
#Selecting Earning and Credit Score column as we want to do prediction on those columns
and assigning to variable Y
Y=loan.iloc[:,[2,3]].values
```

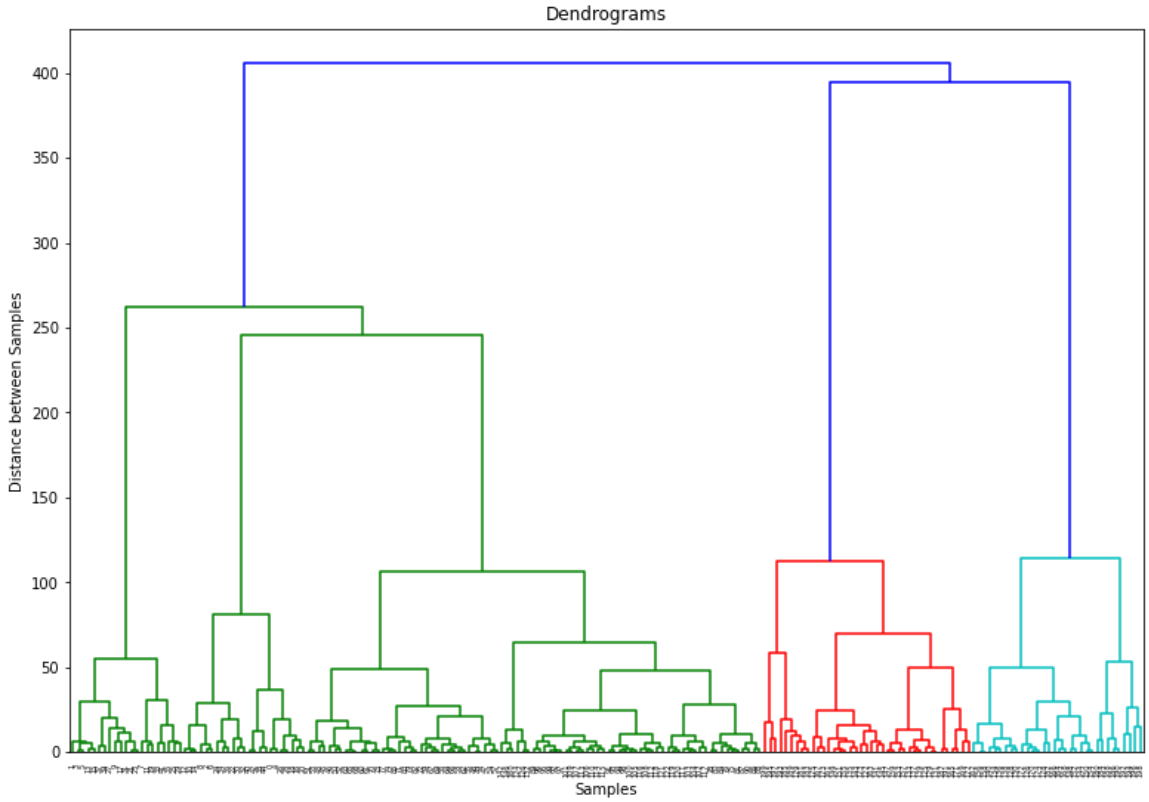
Question-2: Identify the number of clusters using Dendrogram (Intermediate)

In [20]:

```
plt.figure(figsize=(13, 9))
plt.title("Dendrograms")
dendrogram = shc.dendrogram(shc.linkage(Y, method='ward'))#ward is one of the methods t
hat is used to calculate distance between newly formed clusters
plt.xlabel('Samples')
plt.ylabel('Distance between Samples')
```

Out[20]:

```
Text(0, 0.5, 'Distance between Samples')
```



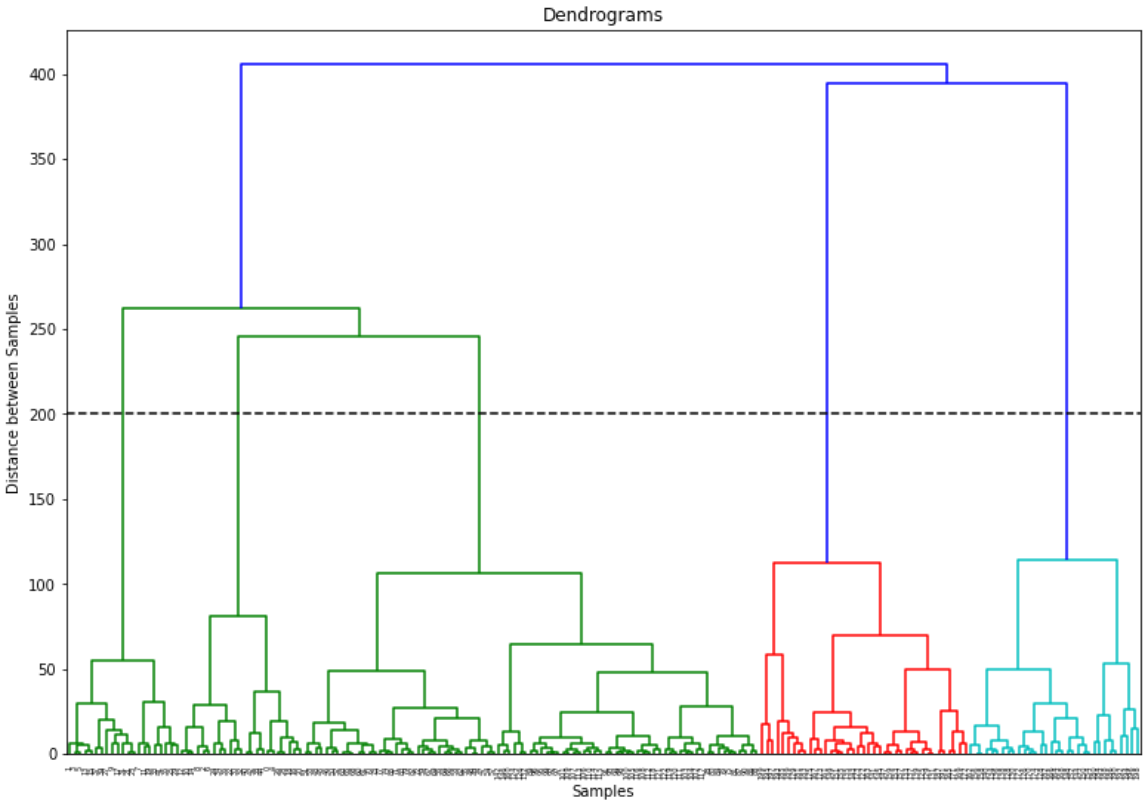
To interpret a dendrogram, one must focus on the height at which any two objects are joined together.

From above, you can see that the blue and green line has the maximum distance. We can select a threshold of 200 and then cut the dendrogram.

Question-3: Apply a threshold value to the above visualization (dendrogram) to find the clusters (Intermediate)

In [21]:

```
plt.figure(figsize=(13, 9))
plt.title("Dendrograms")
dendrogram = shc.dendrogram(shc.linkage(Y, method='ward'))#ward is used to calculate di
stance between newly formed clusters and can only be used with Euclidean Distance
plt.axhline(y=200, color='k', linestyle='--')
plt.xlabel('Samples')
plt.ylabel('Distance between Samples')
plt.show()
```



From the above, you can see that the line cuts the dendrogram at five points. That means we are going to apply hierarchial clustering for five clusters.

Question-4: ¶

Create, fit and predict hierarchical model with five clusters found by Dendrogram (Intermediate)

In [22]:

```
h_clustering=AgglomerativeClustering(n_clusters=5,affinity='euclidean',linkage='ward')
h_clustering1=h_clustering.fit_predict(Y)
print(h_clustering1)
```

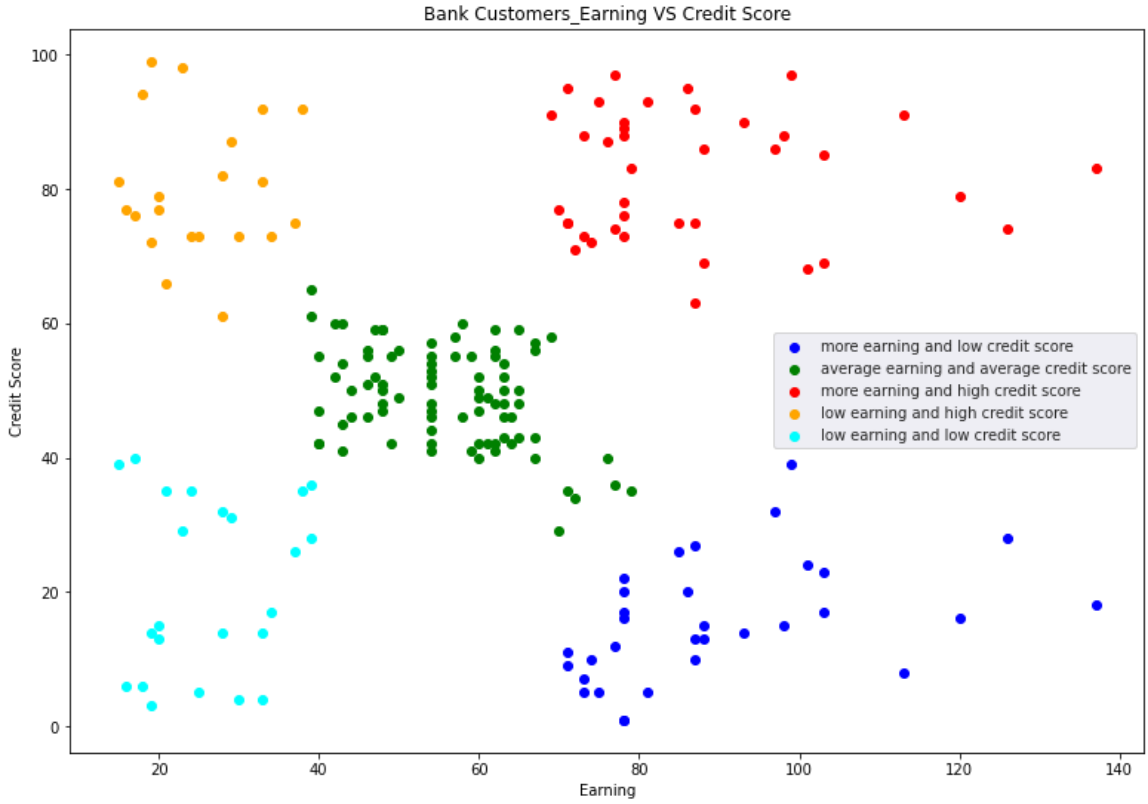
```
[4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
 3 4 3 4 3 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2 1 2
 0 2 0 2 0 2 0 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
```

Question-5:

Visulalize customer based on their ***Earning*** and ***Credit Score*** (Advanced)

In [23]:

```
#Visualizing the clusters using scatter plots
plt.scatter(Y[h_clustering1==0,0],Y[h_clustering1==0,1],color='blue',label='more earnin
g and low credit score')
plt.scatter(Y[h_clustering1==1,0],Y[h_clustering1==1,1],color='green',label='average ea
rning and average credit score')
plt.scatter(Y[h_clustering1==2,0],Y[h_clustering1==2,1],color='red',label='more earning
and high credit score')
plt.scatter(Y[h_clustering1==3,0],Y[h_clustering1==3,1],color='orange',label='low earni
ng and high credit score')
plt.scatter(Y[h_clustering1==4,0],Y[h_clustering1==4,1],color='cyan',label='low earning
and low credit score')
plt.title('Bank Customers_Earning VS Credit Score')
fig=plt.gcf()
fig.set_size_inches(13,9)
sns.set_style('darkgrid')
plt.xlabel('Earning')
plt.ylabel('Credit Score')
plt.legend()
plt.show()
```



We have 5 clusters of Customer based on Earning and Credit Score:

- **Cluster-1 (Blue)** : Customers of this group earn more but maintain a shallow credit score.
- **Cluster-2 (Green)** : Customers of this group are earn average and maintain a decent credit score. Bank may consider them for the new loan scheme
- **Cluster-3 (Red)** : Customers of this group earn more and have a high credit score. Bank will surely consider them for the new loan scheme
- **Cluster-4 (Orange)** : These customers earn less but have a high credit score, seems like they repay their loan on time
- **Cluster-5 (Cyan)** : These customers earn less and have a low credit score, they might not be eligible for the new loan scheme

Module 3 - Association Rule Mining

- An Association Rule **A** -> **B**, where **A** & **B** are two disjoint sets of items

> **A** -> **B** means that if a customer buys item **A**, he/she is also likely to buy item **B**.

- For example, Customers who purchase a **pencil** have a
certain likelihood of purchasing a **sharpener** for their pencil as well.

Understanding Apriori Algorithm

- A classical algorithm used in Data Mining

- Used for finding frequent itemsets and mining association rules that may exist between different itemsets

- Operates very well on a dataset containing a large number of transactions

- Easy to understand and implement

Recommendation Engine

- Recommendation Engine (or Recommender System) predicts what a user may or may not like among a list o

- Helps the user discover products or content that the user may not have come across otherwise

Questions 1-3:

Scenario 1 - Analysing Biscuits Dataset

Dataset Description

The dataset contains entries of Biscuits purchased by Customers at a Supermarket in Bengaluru at any given point.

```
dataset = [['Hide & Seek', 'Krack Jack', 'Bourbon', 'Jim Jam', 'Milk Bikis', 'Da
rk Fantasy', 'Oreo', 'Pencil'],
           ['Hide & Seek','Bourbon','Oreo', 'Unibic', 'Krack Jack','Dark Fantas
y'],
           ['Hide & Seek','Bourbon','Jim Jam','Milk Bikis','Dark Fantasy'],
           ['Hide & Seek', 'Bourbon','Jim Jam' 'Yogurt', 'Milk Bikis','Dark Fant
asy'],
           ['Bourbon', 'Milk Bikis', 'Unibic', 'Oreo', 'Krack Jack'],
           ['Oreo','Unibic','Krack Jack' 'Milk Bikis','Jim Jam'],
           ['Hide & Seek','Milk Bikis','Dark Fantasy'],
           ['Bourbon','Jim Jam','Milk Bikis','Dark Fantasy']]
```

Let us say that the supermarket owner hired you and wants you to use this dataset to help him increase his sales by extracting hidden insights from the dataset.

Tasks to be Performed

- Import the required libraries and transform the dataset using the **TransactionEncoder( )** - Beginner
- Implement the Apriori Algorithm on the **Biscuits** dataset - Beginner
- Mine the **Association Rules** from the **frequent_itemsets** generated previously - Advanced

Topics Covered

- Association Rule Mining
- Apriori Algorithm

Question 1 - Beginner

Import the required libraries and transform the dataset using the **TransactionEncoder( )**.

In [ ]:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import warnings
warnings.filterwarnings("ignore")

dataset = [['Hide & Seek', 'Krack Jack', 'Bourbon', 'Jim Jam', 'Milk Bikis', 'Dark Fantasy', 'Oreo', 'Pencil'],
           ['Hide & Seek','Bourbon','Oreo', 'Unibic', 'Krack Jack','Dark Fantasy'],
           ['Hide & Seek','Bourbon','Jim Jam','Milk Bikis','Dark Fantasy'],
           ['Hide & Seek', 'Bourbon','Jim Jam' 'Yogurt', 'Milk Bikis','Dark Fantasy'],
           ['Bourbon', 'Milk Bikis', 'Unibic', 'Oreo', 'Krack Jack'],
           ['Oreo','Unibic','Krack Jack' 'Milk Bikis','Jim Jam'],
           ['Hide & Seek','Milk Bikis','Dark Fantasy'],
           ['Bourbon','Jim Jam','Milk Bikis','Dark Fantasy']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
df.head()
```

Out[ ]:

	Bourbon	Dark Fantasy	Hide & Seek	Jim Jam	Jim JamYogurt	Krack Jack	Krack JackMilk Bikis	Milk Bikis	Oreo	Pencil	Unibic
0	True	True	True	True	False	True	False	True	True	True	False
1	True	True	True	False	False	True	False	False	True	False	True
2	True	True	True	True	False	False	False	True	False	False	False
3	True	True	True	False	True	False	False	True	False	False	False
4	True	False	False	False	False	True	False	True	True	False	True

Question 2 - Beginner (Bridging Question)

Implement the Apriori Algorithm on the **Biscuits** dataset.

In [ ]:

```
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

frequent_itemsets
```

Out[ ]:

	support	itemsets
0	0.750	(Bourbon)
1	0.750	(Dark Fantasy)
2	0.625	(Hide & Seek)
3	0.750	(Milk Bikis)
4	0.625	(Bourbon, Dark Fantasy)
5	0.625	(Bourbon, Milk Bikis)
6	0.625	(Dark Fantasy, Hide & Seek)
7	0.625	(Dark Fantasy, Milk Bikis)

From the above, you can see that the result is a dataframe with support for each itemsets.

The three major components of Apriori Algorithm are:

- **Support:** It can be defined as the popularity of a particular item. It can be calculated as the number of transactions involving that particular item divided by total number of transactions.
- **Confidence:** It is the likelihood of an item **B** being purchased when **A** was purchased.
- **Lift:** It is the likelihood of an item **B** being purchased when item **A** is purchased while considering the popularity of **B**.

Question 3 - Advanced (Bridging Question)

Mine the **Association Rules** from the **frequent_itemsets** generated previously.

In [ ]:

```
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
0	(Bourbon)	(Dark Fantasy)	0.750	0.750	0.625	0.833333	1.111111	0.062
1	(Dark Fantasy)	(Bourbon)	0.750	0.750	0.625	0.833333	1.111111	0.062
2	(Bourbon)	(Milk Bikis)	0.750	0.750	0.625	0.833333	1.111111	0.062
3	(Milk Bikis)	(Bourbon)	0.750	0.750	0.625	0.833333	1.111111	0.062
4	(Dark Fantasy)	(Hide & Seek)	0.750	0.625	0.625	0.833333	1.333333	0.156
5	(Hide & Seek)	(Dark Fantasy)	0.625	0.750	0.625	1.000000	1.333333	0.156
6	(Dark Fantasy)	(Milk Bikis)	0.750	0.750	0.625	0.833333	1.111111	0.062
7	(Milk Bikis)	(Dark Fantasy)	0.750	0.750	0.625	0.833333	1.111111	0.062

From the above, you can see the result of association analysis showing which item is frequently purchased with other items.

In [ ]:

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
rules
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
0	(Dark Fantasy)	(Hide & Seek)	0.750	0.625	0.625	0.833333	1.333333	0.156
1	(Hide & Seek)	(Dark Fantasy)	0.625	0.750	0.625	1.000000	1.333333	0.156

In [ ]:

```
rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
rules
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
0	(Dark Fantasy)	(Hide & Seek)	0.750	0.625	0.625	0.833333	1.333333	0.156
1	(Hide & Seek)	(Dark Fantasy)	0.625	0.750	0.625	1.000000	1.333333	0.156

In [ ]:

```
rules[ (rules['antecedent_len'] >= 0.5) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
0	(Dark Fantasy)	(Hide & Seek)	0.750	0.625	0.625	0.833333	1.333333	0.156
1	(Hide & Seek)	(Dark Fantasy)	0.625	0.750	0.625	1.000000	1.333333	0.156

In [ ]:

```
rules[rules['antecedents'] == {'Hide & Seek'}]
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
1	(Hide & Seek)	(Dark Fantasy)	0.625	0.75	0.625	1.0	1.333333	0.156

From the above, we can say that when a customer buys **Hide & Seek**, he/she is likely to buy **Dark Fantasy** as well. So, to increase the sales we can put these items together.

Questions 4-6:

Scenario 2 - Analyzing Groceries Dataset

Dataset Description

The dataset contains 9835 transactions with 32 different items being sold at the store over the course of a week.

The Manager of a retail store, located in Chennai is trying to find out an association rule between 32 different items, to find out which items are frequently bought together so that he can keep the items together to boost the sales.

You, as a Data Scientist, are required to perform Market Basket Analysis using Apriori Algorithm on a dataset containing 9835 transactions of a retail store.

Tasks to be Performed

- Load, analyze, and pre-process the dataset and convert the dataframe into a list of lists - Beginner
- Implement the Apriori Algorithm from the Apyori package - Intermediate
- Analyze the first relation obtained from the rules obtained & derive useful insights - Advanced

Topics Covered

- Association Rule Mining
- Apriori Algorithm

In [ ]:

```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/6gnowmpqro12b3o/Grocery.csv
```

```
--2020-06-29 11:42:41--  https://www.dropbox.com/s/6gnowmpqro12b3o/Grocer
y.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:603
2:1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... conne
cted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/6gnowmpqro12b3o/Grocery.csv [following]
--2020-06-29 11:42:41--  https://www.dropbox.com/s/raw/6gnowmpqro12b3o/Gro
cery.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc76ad03b624974c2b8c4220ce44.dl.dropboxusercontent.com/c
d/0/inline/A6l-4BH2I6y0j3NoY1GFI48Pv8WJCRpXOpxA1ZoRF_Lu3XI6wj5FPeNk6QFVIw6
6fSJRMitn67Mb1BMpZR2mAGcnMIIZ2jOFlmfPgFw2-J8a-GowEI_g6QRQ4ot1jzYbLjs/file#
[following]
--2020-06-29 11:42:41--  https://uc76ad03b624974c2b8c4220ce44.dl.dropboxus
ercontent.com/cd/0/inline/A6l-4BH2I6y0j3NoY1GFI48Pv8WJCRpXOpxA1ZoRF_Lu3XI6
wj5FPeNk6QFVIw66fSJRMitn67Mb1BMpZR2mAGcnMIIZ2jOFlmfPgFw2-J8a-GowEI_g6QRQ4o
t1jzYbLjs/file
Resolving uc76ad03b624974c2b8c4220ce44.dl.dropboxusercontent.com (uc76ad03
b624974c2b8c4220ce44.dl.dropboxusercontent.com)... 162.125.82.15, 2620:10
0:6032:15::a27d:520f
Connecting to uc76ad03b624974c2b8c4220ce44.dl.dropboxusercontent.com (uc76
ad03b624974c2b8c4220ce44.dl.dropboxusercontent.com)|162.125.82.15|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 782050 (764K) [text/plain]
Saving to: ‘Grocery.csv’

Grocery.csv         100%[===================>] 763.72K  --.-KB/s    in 0.0
8s

2020-06-29 11:42:42 (9.18 MB/s) - ‘Grocery.csv’ saved [782050/782050]
```

Question 4 - Beginner

Load, analyze, and pre-process the dataset and convert the dataframe into a list of lists.

In [ ]:

```
df = pd.read_csv('/content/Grocery.csv',header=None)
df.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	
0	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
1	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

From the above, you can see that the dataset contains NaN values. That means that item was not purchased in that transaction.

Analyzing the Dataset

In [ ]:

```
df.describe()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	
count	9835	7676	6033	4734	3729	2874	2229	1684	1246	896	(
unique	158	151	155	153	150	137	138	140	128	120	
top	sausage	whole milk	whole milk	whole milk	rolls/buns	soda	soda	shopping bags	soda	shopping bags	shopp b
freq	825	654	506	315	176	150	120	76	61	49	

In [ ]:

```
df.shape
```

Out[ ]:

```
(9835, 32)
```

Converting the dataframe into a list of lists

Apriori library requires the dataset to be in the form of a list of lists, where the entire dataset is a big list and each transaction in the dataset is stored as a list inside the bigger list.

In [ ]:

```
dataset=[]
for i in range(0, df.shape[0]):
    dataset.append([str(df.values[i,j]) for j in range(0, df.shape[1])])
```

Question 5 - Intermediate (Bridging Question)

Implement the Apriori Algorithm from the Apyori package.

In [ ]:

```
!pip install apyori
```

```
Requirement already satisfied: apyori in /usr/local/lib/python3.6/dist-pac
kages (1.1.2)
```

In [ ]:

```
import pandas as pd
import numpy as np
from apyori import apriori
import warnings
warnings.filterwarnings("ignore")

rules = apriori(dataset, min_support=0.002,min_confidence=0.7,min_lift=3,min_length=1,
use_colnames=True)
result = list(rules)

result
```

Out[ ]:

```
[RelationRecord(items=frozenset({'baking powder', 'root vegetables', 'othe
r vegetables'}), support=0.002541942043721403, ordered_statistics=[Ordered
Statistic(items_base=frozenset({'baking powder', 'root vegetables'}), item
s_add=frozenset({'other vegetables'}), confidence=0.7142857142857143, lift
=3.69153674198634)]),
 RelationRecord(items=frozenset({'citrus fruit', 'herbs', 'other vegetable
s'}), support=0.002135231316725979, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'citrus fruit', 'herbs'}), items_add=frozenset({'othe
r vegetables'}), confidence=0.7241379310344829, lift=3.742457462808270
7)]),
 RelationRecord(items=frozenset({'whole milk', 'curd', 'hamburger meat'}),
support=0.002541942043721403, ordered_statistics=[OrderedStatistic(items_b
ase=frozenset({'curd', 'hamburger meat'}), items_add=frozenset({'whole mil
k'}), confidence=0.8064516129032258, lift=3.156168568604546)]),
 RelationRecord(items=frozenset({'herbs', 'whole milk', 'rolls/buns'}), su
pport=0.002440264361972547, ordered_statistics=[OrderedStatistic(items_bas
e=frozenset({'herbs', 'rolls/buns'}), items_add=frozenset({'whole milk'}),
confidence=0.8, lift=3.13091922005571)]),
 RelationRecord(items=frozenset({'herbs', 'tropical fruit', 'whole mil
k'}), support=0.002338586680223691, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'herbs', 'tropical fruit'}), items_add=frozenset({'wh
ole milk'}), confidence=0.8214285714285714, lift=3.2147831277357737)]),
 RelationRecord(items=frozenset({'root vegetables', 'other vegetables', 'r
ice'}), support=0.0022369089984748346, ordered_statistics=[OrderedStatisti
c(items_base=frozenset({'root vegetables', 'rice'}), items_add=frozenset
({'other vegetables'}), confidence=0.7096774193548386, lift=3.667723289203
803)]),
 RelationRecord(items=frozenset({'root vegetables', 'other vegetables', 's
oft cheese'}), support=0.002440264361972547, ordered_statistics=[OrderedSt
atistic(items_base=frozenset({'root vegetables', 'soft cheese'}), items_ad
d=frozenset({'other vegetables'}), confidence=0.7272727272727273, lift=3.7
58658577365882)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'soft cheese', 'oth
er vegetables'}), support=0.0022369089984748346, ordered_statistics=[Order
edStatistic(items_base=frozenset({'whipped/sour cream', 'soft cheese'}), i
tems_add=frozenset({'other vegetables'}), confidence=0.7333333333333333, l
ift=3.7899807321772636)]),
 RelationRecord(items=frozenset({'whole milk', 'root vegetables', 'ric
e'}), support=0.002440264361972547, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'root vegetables', 'rice'}), items_add=frozenset({'wh
ole milk'}), confidence=0.7741935483870968, lift=3.0299218258603644)]),
 RelationRecord(items=frozenset({'baking powder', 'other vegetables', 'roo
t vegetables', 'nan'}), support=0.002541942043721403, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'baking powder', 'root vegetable
s'}), items_add=frozenset({'other vegetables', 'nan'}), confidence=0.71428
57142857143, lift=3.6934805467928493), OrderedStatistic(items_base=frozens
et({'baking powder', 'root vegetables', 'nan'}), items_add=frozenset({'oth
er vegetables'}), confidence=0.7142857142857143, lift=3.69153967419863
4)]),
 RelationRecord(items=frozenset({'root vegetables', 'beef', 'other vegetab
les', 'tropical fruit'}), support=0.0027452974072191155, ordered_statistic
s=[OrderedStatistic(items_base=frozenset({'root vegetables', 'beef', 'trop
ical fruit'}), items_add=frozenset({'other vegetables'}), confidence=0.729
7297297297297, lift=3.7713567482353607)]),
 RelationRecord(items=frozenset({'beef', 'rolls/buns', 'tropical fruit',
'whole milk'}), support=0.002135231316725979, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'beef', 'rolls/buns', 'tropical fruit'}), i
tems_add=frozenset({'whole milk'}), confidence=0.7777777777777778, lift=3.
043949241720829)]),
```

```
 RelationRecord(items=frozenset({'domestic eggs', 'whole milk', 'other veg
etables', 'bottled beer'}), support=0.0020335536349771225, ordered_statist
ics=[OrderedStatistic(items_base=frozenset({'domestic eggs', 'other vegeta
bles', 'bottled beer'}), items_add=frozenset({'whole milk'}), confidence=
0.7692307692307693, lift=3.0104992500535674)]),
 RelationRecord(items=frozenset({'brown bread', 'root vegetables', 'other
vegetables', 'whole milk'}), support=0.00315200813421454, ordered_statisti
cs=[OrderedStatistic(items_base=frozenset({'brown bread', 'root vegetable
s', 'other vegetables'}), items_add=frozenset({'whole milk'}), confidence=
0.775, lift=3.0330779944289694)]),
 RelationRecord(items=frozenset({'whole milk', 'curd', 'yogurt', 'butte
r'}), support=0.002338586680223691, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'yogurt', 'curd', 'butter'}), items_add=frozenset({'w
hole milk'}), confidence=0.7931034482758621, lift=3.1039285371241956)]),
 RelationRecord(items=frozenset({'whole milk', 'other vegetables', 'pork',
'butter'}), support=0.0022369089984748346, ordered_statistics=[OrderedStat
istic(items_base=frozenset({'other vegetables', 'pork', 'butter'}), items_
add=frozenset({'whole milk'}), confidence=0.8461538461538461, lift=3.31154
9175058924)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'other vegetables',
'tropical fruit', 'butter'}), support=0.002338586680223691, ordered_statis
tics=[OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'tropic
al fruit', 'butter'}), items_add=frozenset({'other vegetables'}), confiden
ce=0.7666666666666666, lift=3.9622525836398665)]),
 RelationRecord(items=frozenset({'whole milk', 'root vegetables', 'yogur
t', 'butter'}), support=0.003050330452465684, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'yogurt', 'root vegetables', 'butter'}), it
ems_add=frozenset({'whole milk'}), confidence=0.7894736842105263, lift=3.0
897229145286613)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'other
vegetables', 'frozen vegetables'}), support=0.0020335536349771225, ordered
_statistics=[OrderedStatistic(items_base=frozenset({'citrus fruit', 'root
vegetables', 'frozen vegetables'}), items_add=frozenset({'other vegetable
s'}), confidence=0.7692307692307693, lift=3.975504264521606)]),
 RelationRecord(items=frozenset({'other vegetables', 'citrus fruit', 'herb
s', 'nan'}), support=0.002135231316725979, ordered_statistics=[OrderedStat
istic(items_base=frozenset({'citrus fruit', 'herbs'}), items_add=frozenset
({'other vegetables', 'nan'}), confidence=0.7241379310344829, lift=3.74442
5106058958), OrderedStatistic(items_base=frozenset({'citrus fruit', 'herb
s', 'nan'}), items_add=frozenset({'other vegetables'}), confidence=0.72413
79310344829, lift=3.7424574628082707)]),
 RelationRecord(items=frozenset({'soda', 'citrus fruit', 'root vegetable
s', 'other vegetables'}), support=0.002135231316725979, ordered_statistics
=[OrderedStatistic(items_base=frozenset({'soda', 'citrus fruit', 'root veg
etables'}), items_add=frozenset({'other vegetables'}), confidence=0.700000
0000000001, lift=3.6177088807146616)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'other
vegetables', 'tropical fruit'}), support=0.004473817996949669, ordered_sta
tistics=[OrderedStatistic(items_base=frozenset({'citrus fruit', 'root vege
tables', 'tropical fruit'}), items_add=frozenset({'other vegetables'}), co
nfidence=0.7857142857142856, lift=4.060693641618497)]),
 RelationRecord(items=frozenset({'domestic eggs', 'curd', 'other vegetable
s', 'whole milk'}), support=0.0028469750889679717, ordered_statistics=[Ord
eredStatistic(items_base=frozenset({'domestic eggs', 'curd', 'other vegeta
bles'}), items_add=frozenset({'whole milk'}), confidence=0.82352941176470
6, lift=3.2230050794691136)]),
 RelationRecord(items=frozenset({'whole milk', 'curd', 'nan', 'hamburger m
eat'}), support=0.002541942043721403, ordered_statistics=[OrderedStatistic
(items_base=frozenset({'curd', 'hamburger meat'}), items_add=frozenset({'n
an', 'whole milk'}), confidence=0.8064516129032258, lift=3.157425005136634
6), OrderedStatistic(items_base=frozenset({'curd', 'nan', 'hamburger mea
```

```
t'}), items_add=frozenset({'whole milk'}), confidence=0.8064516129032258,
lift=3.156168568604546)]),
 RelationRecord(items=frozenset({'domestic eggs', 'root vegetables', 'trop
ical fruit', 'whole milk'}), support=0.0027452974072191155, ordered_statis
tics=[OrderedStatistic(items_base=frozenset({'domestic eggs', 'root vegeta
bles', 'tropical fruit'}), items_add=frozenset({'whole milk'}), confidence
=0.7714285714285715, lift=3.019100676482292)]),
 RelationRecord(items=frozenset({'root vegetables', 'frozen vegetables',
'whole milk', 'tropical fruit'}), support=0.002338586680223691, ordered_st
atistics=[OrderedStatistic(items_base=frozenset({'root vegetables', 'froze
n vegetables', 'tropical fruit'}), items_add=frozenset({'whole milk'}), co
nfidence=0.7666666666666666, lift=3.0004642525533884)]),
 RelationRecord(items=frozenset({'other vegetables', 'root vegetables', 'f
ruit/vegetable juice', 'tropical fruit'}), support=0.002541942043721403, o
rdered_statistics=[OrderedStatistic(items_base=frozenset({'root vegetable
s', 'fruit/vegetable juice', 'tropical fruit'}), items_add=frozenset({'oth
er vegetables'}), confidence=0.78125, lift=4.037621518654755)]),
 RelationRecord(items=frozenset({'yogurt', 'other vegetables', 'root veget
ables', 'fruit/vegetable juice'}), support=0.002440264361972547, ordered_s
tatistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root vegetabl
es', 'fruit/vegetable juice'}), items_add=frozenset({'other vegetables'}),
confidence=0.7058823529411765, lift=3.64810979567865)]),
 RelationRecord(items=frozenset({'other vegetables', 'grapes', 'whole mil
k', 'tropical fruit'}), support=0.0020335536349771225, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'grapes', 'whole milk', 'tropical
fruit'}), items_add=frozenset({'other vegetables'}), confidence=0.8, lift=
4.13452443510247)]),
 RelationRecord(items=frozenset({'whole milk', 'herbs', 'nan', 'rolls/bun
s'}), support=0.002440264361972547, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'herbs', 'rolls/buns'}), items_add=frozenset({'nan',
'whole milk'}), confidence=0.8, lift=3.1321656050955418), OrderedStatistic
(items_base=frozenset({'herbs', 'nan', 'rolls/buns'}), items_add=frozenset
({'whole milk'}), confidence=0.8, lift=3.13091922005571)]),
 RelationRecord(items=frozenset({'whole milk', 'herbs', 'tropical fruit',
'nan'}), support=0.002338586680223691, ordered_statistics=[OrderedStatisti
c(items_base=frozenset({'herbs', 'tropical fruit'}), items_add=frozenset
({'nan', 'whole milk'}), confidence=0.8214285714285714, lift=3.21606289808
91723), OrderedStatistic(items_base=frozenset({'herbs', 'tropical fruit',
'nan'}), items_add=frozenset({'whole milk'}), confidence=0.821428571428571
4, lift=3.2147831277357737)]),
 RelationRecord(items=frozenset({'other vegetables', 'root vegetables', 'n
an', 'rice'}), support=0.0022369089984748346, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'root vegetables', 'rice'}), items_add=froz
enset({'other vegetables', 'nan'}), confidence=0.7096774193548386, lift=3.
669651640039347), OrderedStatistic(items_base=frozenset({'root vegetable
s', 'nan', 'rice'}), items_add=frozenset({'other vegetables'}), confidence
=0.7096774193548386, lift=3.667723289203803)]),
 RelationRecord(items=frozenset({'other vegetables', 'root vegetables', 'n
an', 'soft cheese'}), support=0.002338586680223691, ordered_statistics=[Or
deredStatistic(items_base=frozenset({'root vegetables', 'nan', 'soft chees
e'}), items_add=frozenset({'other vegetables'}), confidence=0.71875, lift=
3.7146117971623753)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'other vegetables',
'nan', 'soft cheese'}), support=0.002135231316725979, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'soft chees
e'}), items_add=frozenset({'other vegetables', 'nan'}), confidence=0.70000
00000000001, lift=3.6196109358569926), OrderedStatistic(items_base=frozens
et({'whipped/sour cream', 'soft cheese', 'nan'}), items_add=frozenset({'ot
her vegetables'}), confidence=0.7241379310344829, lift=3.742457462808270
7)]),
 RelationRecord(items=frozenset({'whole milk', 'root vegetables', 'nan',
```

'rice'}), support=0.002440264361972547, ordered_statistics=[OrderedStatist
ic(items_base=frozenset({'root vegetables', 'rice'}), items_add=frozenset
({'nan', 'whole milk'}), confidence=0.7741935483870968, lift=3.03112800493
11695), OrderedStatistic(items_base=frozenset({'root vegetables', 'nan',
'rice'}), items_add=frozenset({'whole milk'}), confidence=0.77419354838709
68, lift=3.0299218258603644)]),
 RelationRecord(items=frozenset({'yogurt', 'oil', 'other vegetables', 'who
le milk'}), support=0.0022369089984748346, ordered_statistics=[OrderedStat
istic(items_base=frozenset({'yogurt', 'oil', 'whole milk'}), items_add=fro
zenset({'other vegetables'}), confidence=0.7096774193548386, lift=3.667723
289203803)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'other vegetables',
'whole milk', 'onions'}), support=0.002135231316725979, ordered_statistics
=[OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'whole mil
k', 'onions'}), items_add=frozenset({'other vegetables'}), confidence=0.77
77777777777778, lift=4.0196765341274014)]),
 RelationRecord(items=frozenset({'pip fruit', 'other vegetables', 'pork',
'whole milk'}), support=0.002338586680223691, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'pip fruit', 'pork', 'whole milk'}), items_
add=frozenset({'other vegetables'}), confidence=0.7419354838709677, lift=
3.834437984167613)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'pip fruit', 'other
vegetables', 'tropical fruit'}), support=0.0027452974072191155, ordered_st
atistics=[OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'pi
p fruit', 'tropical fruit'}), items_add=frozenset({'other vegetables'}), c
onfidence=0.7297297297297297, lift=3.7713567482353607)]),
 RelationRecord(items=frozenset({'yogurt', 'root vegetables', 'other veget
ables', 'shopping bags'}), support=0.002135231316725979, ordered_statistic
s=[OrderedStatistic(items_base=frozenset({'yogurt', 'root vegetables', 'sh
opping bags'}), items_add=frozenset({'other vegetables'}), confidence=0.72
41379310344829, lift=3.7424574628082707)]),
 RelationRecord(items=frozenset({'root vegetables', 'other vegetables', 'w
hole milk', 'sliced cheese'}), support=0.002440264361972547, ordered_stati
stics=[OrderedStatistic(items_base=frozenset({'root vegetables', 'sliced c
heese', 'whole milk'}), items_add=frozenset({'other vegetables'}), confide
nce=0.7741935483870968, lift=4.0011526791314225)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'root vegetables',
'other vegetables', 'tropical fruit'}), support=0.003355363497712252, orde
red_statistics=[OrderedStatistic(items_base=frozenset({'whipped/sour crea
m', 'root vegetables', 'tropical fruit'}), items_add=frozenset({'other veg
etables'}), confidence=0.7333333333333333, lift=3.7899807321772636)]),
 RelationRecord(items=frozenset({'whipped/sour cream', 'root vegetables',
'pip fruit', 'whole milk'}), support=0.002338586680223691, ordered_statist
ics=[OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'root ve
getables', 'pip fruit'}), items_add=frozenset({'whole milk'}), confidence=
0.7666666666666666, lift=3.0004642525533884)]),
 RelationRecord(items=frozenset({'sausage', 'root vegetables', 'tropical f
ruit', 'whole milk'}), support=0.0027452974072191155, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'sausage', 'root vegetables', 'tro
pical fruit'}), items_add=frozenset({'whole milk'}), confidence=0.77142857
14285715, lift=3.019100676482292)]),
 RelationRecord(items=frozenset({'root vegetables', 'beef', 'other vegetab
les', 'nan', 'tropical fruit'}), support=0.0026436197254702592, ordered_st
atistics=[OrderedStatistic(items_base=frozenset({'root vegetables', 'bee
f', 'tropical fruit'}), items_add=frozenset({'other vegetables', 'nan'}),
confidence=0.7027027027027026, lift=3.6335862676556676), OrderedStatistic
(items_base=frozenset({'root vegetables', 'beef', 'nan', 'tropical frui
t'}), items_add=frozenset({'other vegetables'}), confidence=0.722222222222
2222, lift=3.7325567816897296)]),
 RelationRecord(items=frozenset({'whole milk', 'beef', 'nan', 'rolls/bun
s', 'tropical fruit'}), support=0.0020335536349771225, ordered_statistics=

[OrderedStatistic(items_base=frozenset({'beef', 'nan', 'tropical fruit',
'rolls/buns'}), items_add=frozenset({'whole milk'}), confidence=0.76923076
92307693, lift=3.0104992500535674)]),
 RelationRecord(items=frozenset({'whole milk', 'bottled beer', 'domestic e
ggs', 'other vegetables', 'nan'}), support=0.0020335536349771225, ordered_
statistics=[OrderedStatistic(items_base=frozenset({'domestic eggs', 'other
vegetables', 'bottled beer'}), items_add=frozenset({'nan', 'whole milk'}),
confidence=0.7692307692307693, lift=3.011697697207252), OrderedStatistic(i
tems_base=frozenset({'domestic eggs', 'other vegetables', 'nan', 'bottled
beer'}), items_add=frozenset({'whole milk'}), confidence=0.769230769230769
3, lift=3.0104992500535674)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'other v
egetables', 'nan', 'brown bread'}), support=0.00315200813421454, ordered_s
tatistics=[OrderedStatistic(items_base=frozenset({'brown bread', 'root veg
etables', 'other vegetables'}), items_add=frozenset({'nan', 'whole mil
k'}), confidence=0.775, lift=3.0342854299363062), OrderedStatistic(items_b
ase=frozenset({'brown bread', 'other vegetables', 'root vegetables', 'na
n'}), items_add=frozenset({'whole milk'}), confidence=0.775, lift=3.033077
9944289694)]),
 RelationRecord(items=frozenset({'whole milk', 'curd', 'butter', 'yogurt',
'nan'}), support=0.0022369089984748346, ordered_statistics=[OrderedStatist
ic(items_base=frozenset({'yogurt', 'curd', 'nan', 'butter'}), items_add=fr
ozenset({'whole milk'}), confidence=0.7857142857142856, lift=3.07500994826
90003)]),
 RelationRecord(items=frozenset({'whole milk', 'other vegetables', 'pork',
'butter', 'nan'}), support=0.0022369089984748346, ordered_statistics=[Orde
redStatistic(items_base=frozenset({'other vegetables', 'pork', 'butter'}),
items_add=frozenset({'nan', 'whole milk'}), confidence=0.8461538461538461,
lift=3.312867466927977), OrderedStatistic(items_base=frozenset({'other veg
etables', 'nan', 'pork', 'butter'}), items_add=frozenset({'whole milk'}),
confidence=0.8461538461538461, lift=3.311549175058924)]),
 RelationRecord(items=frozenset({'other vegetables', 'butter', 'whipped/so
ur cream', 'nan', 'tropical fruit'}), support=0.0022369089984748346, order
ed_statistics=[OrderedStatistic(items_base=frozenset({'whipped/sour crea
m', 'tropical fruit', 'butter'}), items_add=frozenset({'other vegetables',
'nan'}), confidence=0.7333333333333333, lift=3.7919733613739917), OrderedS
tatistic(items_base=frozenset({'whipped/sour cream', 'nan', 'tropical frui
t', 'butter'}), items_add=frozenset({'other vegetables'}), confidence=0.75
86206896551724, lift=3.920669722941997)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'butte
r', 'yogurt', 'nan'}), support=0.0029486527707168276, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'yogurt', 'root vegetables', 'na
n', 'butter'}), items_add=frozenset({'whole milk'}), confidence=0.78378378
37837838, lift=3.067454641270797)]),
 RelationRecord(items=frozenset({'whole milk', 'other vegetables', 'butte
r', 'yogurt', 'tropical fruit'}), support=0.002338586680223691, ordered_st
atistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'other vegetabl
es', 'tropical fruit', 'butter'}), items_add=frozenset({'whole milk'}), co
nfidence=0.7666666666666666, lift=3.0004642525533884)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'other
vegetables', 'frozen vegetables', 'nan'}), support=0.0020335536349771225,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'citrus fruit',
'root vegetables', 'frozen vegetables'}), items_add=frozenset({'other vege
tables', 'nan'}), confidence=0.7692307692307693, lift=3.9775944350076844),
OrderedStatistic(items_base=frozenset({'citrus fruit', 'root vegetables',
'nan', 'frozen vegetables'}), items_add=frozenset({'other vegetables'}), c
onfidence=0.7692307692307693, lift=3.975504264521606)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'other
vegetables', 'nan', 'soda'}), support=0.002135231316725979, ordered_statis
tics=[OrderedStatistic(items_base=frozenset({'soda', 'citrus fruit', 'root
vegetables'}), items_add=frozenset({'other vegetables', 'nan'}), confidenc

e=0.7000000000000001, lift=3.6196109358569926), OrderedStatistic(items_bas
e=frozenset({'soda', 'citrus fruit', 'root vegetables', 'nan'}), items_add
=frozenset({'other vegetables'}), confidence=0.7000000000000001, lift=3.61
77088807146616)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'other
vegetables', 'nan', 'tropical fruit'}), support=0.004372140315200813, orde
red_statistics=[OrderedStatistic(items_base=frozenset({'citrus fruit', 'ro
ot vegetables', 'tropical fruit'}), items_add=frozenset({'other vegetable
s', 'nan'}), confidence=0.7678571428571428, lift=3.9704915878023126), Orde
redStatistic(items_base=frozenset({'citrus fruit', 'root vegetables', 'na
n', 'tropical fruit'}), items_add=frozenset({'other vegetables'}), confide
nce=0.7818181818181819, lift=4.040557970668323)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'whole
milk', 'other vegetables', 'tropical fruit'}), support=0.0031520081342145
4, ordered_statistics=[OrderedStatistic(items_base=frozenset({'citrus frui
t', 'root vegetables', 'tropical fruit', 'whole milk'}), items_add=frozens
et({'other vegetables'}), confidence=0.8857142857142858, lift=4.5775091960
06306)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'whole
milk', 'other vegetables', 'yogurt'}), support=0.002338586680223691, order
ed_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'citrus fr
uit', 'root vegetables', 'other vegetables'}), items_add=frozenset({'whole
milk'}), confidence=0.8214285714285714, lift=3.2147831277357737), OrderedS
tatistic(items_base=frozenset({'yogurt', 'citrus fruit', 'root vegetable
s', 'whole milk'}), items_add=frozenset({'other vegetables'}), confidence=
0.7419354838709677, lift=3.834437984167613)]),
 RelationRecord(items=frozenset({'citrus fruit', 'whole milk', 'other vege
tables', 'yogurt', 'tropical fruit'}), support=0.002440264361972547, order
ed_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'citrus fr
uit', 'other vegetables', 'tropical fruit'}), items_add=frozenset({'whole
milk'}), confidence=0.7741935483870968, lift=3.0299218258603644)]),
 RelationRecord(items=frozenset({'whole milk', 'domestic eggs', 'curd', 'o
ther vegetables', 'nan'}), support=0.0027452974072191155, ordered_statisti
cs=[OrderedStatistic(items_base=frozenset({'domestic eggs', 'curd', 'other
vegetables'}), items_add=frozenset({'nan', 'whole milk'}), confidence=0.79
41176470588235, lift=3.1091349756463096), OrderedStatistic(items_base=froz
enset({'domestic eggs', 'other vegetables', 'curd', 'nan'}), items_add=fro
zenset({'whole milk'}), confidence=0.8181818181818182, lift=3.202076475056
9763)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'frozen
vegetables', 'nan', 'tropical fruit'}), support=0.002338586680223691, orde
red_statistics=[OrderedStatistic(items_base=frozenset({'root vegetables',
'frozen vegetables', 'tropical fruit'}), items_add=frozenset({'nan', 'whol
e milk'}), confidence=0.7666666666666666, lift=3.0016587048832273), Ordere
dStatistic(items_base=frozenset({'root vegetables', 'nan', 'frozen vegetab
les', 'tropical fruit'}), items_add=frozenset({'whole milk'}), confidence=
0.7666666666666666, lift=3.0004642525533884)]),
 RelationRecord(items=frozenset({'root vegetables', 'other vegetables', 'f
ruit/vegetable juice', 'nan', 'tropical fruit'}), support=0.00254194204372
1403, ordered_statistics=[OrderedStatistic(items_base=frozenset({'root veg
etables', 'fruit/vegetable juice', 'tropical fruit'}), items_add=frozenset
({'other vegetables', 'nan'}), confidence=0.78125, lift=4.03974434805467
9), OrderedStatistic(items_base=frozenset({'root vegetables', 'fruit/veget
able juice', 'tropical fruit', 'nan'}), items_add=frozenset({'other vegeta
bles'}), confidence=0.78125, lift=4.037621518654755)]),
 RelationRecord(items=frozenset({'root vegetables', 'other vegetables', 'f
ruit/vegetable juice', 'yogurt', 'nan'}), support=0.002440264361972547, or
dered_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root v
egetables', 'fruit/vegetable juice'}), items_add=frozenset({'other vegetab
les', 'nan'}), confidence=0.7058823529411765, lift=3.6500278344776396), Or
deredStatistic(items_base=frozenset({'yogurt', 'root vegetables', 'fruit/v

```
egetable juice', 'nan'}), items_add=frozenset({'other vegetables'}), confi
dence=0.7058823529411765, lift=3.64810979567865)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'other v
egetables', 'fruit/vegetable juice', 'yogurt'}), support=0.002033553634977
1225, ordered_statistics=[OrderedStatistic(items_base=frozenset({'yogurt',
'other vegetables', 'root vegetables', 'fruit/vegetable juice'}), items_ad
d=frozenset({'whole milk'}), confidence=0.8333333333333333, lift=3.2613741
87558031), OrderedStatistic(items_base=frozenset({'yogurt', 'root vegetabl
es', 'fruit/vegetable juice', 'whole milk'}), items_add=frozenset({'other
vegetables'}), confidence=0.8, lift=4.13452443510247)]),
 RelationRecord(items=frozenset({'whole milk', 'other vegetables', 'grape
s', 'nan', 'tropical fruit'}), support=0.0020335536349771225, ordered_stat
istics=[OrderedStatistic(items_base=frozenset({'grapes', 'whole milk', 'tr
opical fruit'}), items_add=frozenset({'other vegetables', 'nan'}), confide
nce=0.8, lift=4.136698212407992), OrderedStatistic(items_base=frozenset
({'nan', 'grapes', 'whole milk', 'tropical fruit'}), items_add=frozenset
({'other vegetables'}), confidence=0.8, lift=4.13452443510247)]),
 RelationRecord(items=frozenset({'oil', 'whole milk', 'other vegetables',
'yogurt', 'nan'}), support=0.0022369089984748346, ordered_statistics=[Orde
redStatistic(items_base=frozenset({'yogurt', 'oil', 'whole milk'}), items_
add=frozenset({'other vegetables', 'nan'}), confidence=0.7096774193548386,
lift=3.669651640039347), OrderedStatistic(items_base=frozenset({'yogurt',
'oil', 'nan', 'whole milk'}), items_add=frozenset({'other vegetables'}), c
onfidence=0.7096774193548386, lift=3.667723289203803)]),
 RelationRecord(items=frozenset({'whole milk', 'other vegetables', 'whippe
d/sour cream', 'nan', 'onions'}), support=0.002135231316725979, ordered_st
atistics=[OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'wh
ole milk', 'onions'}), items_add=frozenset({'other vegetables', 'nan'}), c
onfidence=0.7777777777777778, lift=4.021789928729992), OrderedStatistic(it
ems_base=frozenset({'whipped/sour cream', 'nan', 'whole milk', 'onions'}),
items_add=frozenset({'other vegetables'}), confidence=0.7777777777777778,
lift=4.0196765341274014)]),
 RelationRecord(items=frozenset({'whole milk', 'pip fruit', 'other vegetab
les', 'pork', 'nan'}), support=0.002338586680223691, ordered_statistics=[O
rderedStatistic(items_base=frozenset({'pip fruit', 'pork', 'whole milk'}),
items_add=frozenset({'other vegetables', 'nan'}), confidence=0.74193548387
09677, lift=3.836453987313863), OrderedStatistic(items_base=frozenset({'pi
p fruit', 'nan', 'pork', 'whole milk'}), items_add=frozenset({'other veget
ables'}), confidence=0.7419354838709677, lift=3.834437984167613)]),
 RelationRecord(items=frozenset({'pip fruit', 'other vegetables', 'whippe
d/sour cream', 'nan', 'tropical fruit'}), support=0.0027452974072191155, o
rdered_statistics=[OrderedStatistic(items_base=frozenset({'whipped/sour cr
eam', 'pip fruit', 'tropical fruit'}), items_add=frozenset({'other vegetab
les', 'nan'}), confidence=0.7297297297297297, lift=3.7733395856424243), Or
deredStatistic(items_base=frozenset({'whipped/sour cream', 'pip fruit', 'n
an', 'tropical fruit'}), items_add=frozenset({'other vegetables'}), confid
ence=0.7297297297297297, lift=3.7713567482353607)]),
 RelationRecord(items=frozenset({'root vegetables', 'shopping bags', 'othe
r vegetables', 'yogurt', 'nan'}), support=0.002135231316725979, ordered_st
atistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root vegetable
s', 'shopping bags'}), items_add=frozenset({'other vegetables', 'nan'}), c
onfidence=0.7241379310344829, lift=3.744425106058958), OrderedStatistic(it
ems_base=frozenset({'yogurt', 'root vegetables', 'nan', 'shopping bags'}),
items_add=frozenset({'other vegetables'}), confidence=0.7241379310344829,
lift=3.7424574628082707)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'other v
egetables', 'nan', 'sliced cheese'}), support=0.002440264361972547, ordere
d_statistics=[OrderedStatistic(items_base=frozenset({'root vegetables', 's
liced cheese', 'whole milk'}), items_add=frozenset({'other vegetables', 'n
an'}), confidence=0.7741935483870968, lift=4.003256334588379), OrderedStat
istic(items_base=frozenset({'root vegetables', 'nan', 'whole milk', 'slice
```

```
d cheese'}), items_add=frozenset({'other vegetables'}), confidence=0.77419
35483870968, lift=4.0011526791314225)]),
 RelationRecord(items=frozenset({'root vegetables', 'other vegetables', 'w
hipped/sour cream', 'nan', 'tropical fruit'}), support=0.00325368581596339
6, ordered_statistics=[OrderedStatistic(items_base=frozenset({'whipped/sou
r cream', 'root vegetables', 'tropical fruit'}), items_add=frozenset({'oth
er vegetables', 'nan'}), confidence=0.711111111111111, lift=3.677065077695
9917), OrderedStatistic(items_base=frozenset({'whipped/sour cream', 'root
vegetables', 'nan', 'tropical fruit'}), items_add=frozenset({'other vegeta
bles'}), confidence=0.7272727272727273, lift=3.758658577365882)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'pip fru
it', 'whipped/sour cream', 'nan'}), support=0.002338586680223691, ordered_
statistics=[OrderedStatistic(items_base=frozenset({'whipped/sour cream',
'root vegetables', 'pip fruit'}), items_add=frozenset({'nan', 'whole mil
k'}), confidence=0.7666666666666666, lift=3.0016587048832273), OrderedStat
istic(items_base=frozenset({'whipped/sour cream', 'root vegetables', 'pip
fruit', 'nan'}), items_add=frozenset({'whole milk'}), confidence=0.7666666
666666666, lift=3.0004642525533884)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'pip fru
it', 'other vegetables', 'tropical fruit'}), support=0.002440264361972547,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'root vegetable
s', 'pip fruit', 'tropical fruit', 'whole milk'}), items_add=frozenset({'o
ther vegetables'}), confidence=0.7741935483870968, lift=4.001152679131422
5)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'pip fru
it', 'other vegetables', 'yogurt'}), support=0.002338586680223691, ordered
_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root vegeta
bles', 'pip fruit', 'other vegetables'}), items_add=frozenset({'whole mil
k'}), confidence=0.7931034482758621, lift=3.1039285371241956)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'yogur
t', 'rolls/buns', 'tropical fruit'}), support=0.0022369089984748346, order
ed_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root vege
tables', 'rolls/buns', 'tropical fruit'}), items_add=frozenset({'whole mil
k'}), confidence=0.8148148148148148, lift=3.188899205612297)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'whole
milk', 'other vegetables', 'nan', 'tropical fruit'}), support=0.0030503304
52465684, ordered_statistics=[OrderedStatistic(items_base=frozenset({'citr
us fruit', 'root vegetables', 'tropical fruit', 'whole milk'}), items_add=
frozenset({'other vegetables', 'nan'}), confidence=0.8571428571428572, lif
t=4.4321766561514195), OrderedStatistic(items_base=frozenset({'citrus frui
t', 'root vegetables', 'whole milk', 'nan', 'tropical fruit'}), items_add=
frozenset({'other vegetables'}), confidence=0.8823529411764706, lift=4.560
137244598312)]),
 RelationRecord(items=frozenset({'citrus fruit', 'root vegetables', 'whole
milk', 'other vegetables', 'yogurt', 'nan'}), support=0.002236908998474834
6, ordered_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'c
itrus fruit', 'root vegetables', 'other vegetables'}), items_add=frozenset
({'nan', 'whole milk'}), confidence=0.7857142857142856, lift=3.07623407643
31206), OrderedStatistic(items_base=frozenset({'yogurt', 'citrus fruit',
'root vegetables', 'whole milk'}), items_add=frozenset({'other vegetable
s', 'nan'}), confidence=0.7096774193548386, lift=3.669651640039347), Order
edStatistic(items_base=frozenset({'citrus fruit', 'root vegetables', 'othe
r vegetables', 'yogurt', 'nan'}), items_add=frozenset({'whole milk'}), con
fidence=0.8148148148148148, lift=3.188899205612297), OrderedStatistic(item
s_base=frozenset({'citrus fruit', 'root vegetables', 'whole milk', 'yogur
t', 'nan'}), items_add=frozenset({'other vegetables'}), confidence=0.73333
33333333333, lift=3.7899807321772636)]),
 RelationRecord(items=frozenset({'citrus fruit', 'whole milk', 'other vege
tables', 'yogurt', 'nan', 'tropical fruit'}), support=0.00233858668022369
1, ordered_statistics=[OrderedStatistic(items_base=frozenset({'citrus frui
t', 'other vegetables', 'yogurt', 'nan', 'tropical fruit'}), items_add=fro
```

```
zenset({'whole milk'}), confidence=0.7666666666666666, lift=3.000464252553
3884)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'other v
egetables', 'fruit/vegetable juice', 'yogurt', 'nan'}), support=0.00203355
36349771225, ordered_statistics=[OrderedStatistic(items_base=frozenset({'y
ogurt', 'other vegetables', 'root vegetables', 'fruit/vegetable juice'}),
items_add=frozenset({'nan', 'whole milk'}), confidence=0.8333333333333333,
lift=3.262672505307856), OrderedStatistic(items_base=frozenset({'yogurt',
'root vegetables', 'fruit/vegetable juice', 'whole milk'}), items_add=froz
enset({'other vegetables', 'nan'}), confidence=0.8, lift=4.13669821240799
2), OrderedStatistic(items_base=frozenset({'root vegetables', 'other veget
ables', 'fruit/vegetable juice', 'yogurt', 'nan'}), items_add=frozenset
({'whole milk'}), confidence=0.8333333333333333, lift=3.261374187558031),
OrderedStatistic(items_base=frozenset({'root vegetables', 'whole milk', 'f
ruit/vegetable juice', 'yogurt', 'nan'}), items_add=frozenset({'other vege
tables'}), confidence=0.8, lift=4.13452443510247)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'pip fru
it', 'other vegetables', 'nan', 'tropical fruit'}), support=0.002440264361
972547, ordered_statistics=[OrderedStatistic(items_base=frozenset({'root v
egetables', 'pip fruit', 'tropical fruit', 'whole milk'}), items_add=froze
nset({'other vegetables', 'nan'}), confidence=0.7741935483870968, lift=4.0
03256334588379), OrderedStatistic(items_base=frozenset({'root vegetables',
'whole milk', 'pip fruit', 'nan', 'tropical fruit'}), items_add=frozenset
({'other vegetables'}), confidence=0.7741935483870968, lift=4.001152679131
4225)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'pip fru
it', 'other vegetables', 'yogurt', 'nan'}), support=0.002338586680223691,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root
vegetables', 'pip fruit', 'other vegetables'}), items_add=frozenset({'na
n', 'whole milk'}), confidence=0.7931034482758621, lift=3.10516417746540
8), OrderedStatistic(items_base=frozenset({'root vegetables', 'pip fruit',
'other vegetables', 'yogurt', 'nan'}), items_add=frozenset({'whole mil
k'}), confidence=0.7931034482758621, lift=3.1039285371241956)]),
 RelationRecord(items=frozenset({'root vegetables', 'whole milk', 'yogur
t', 'nan', 'rolls/buns', 'tropical fruit'}), support=0.002135231316725979,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'yogurt', 'root
vegetables', 'rolls/buns', 'tropical fruit'}), items_add=frozenset({'nan',
'whole milk'}), confidence=0.7777777777777778, lift=3.0451610049539988), O
rderedStatistic(items_base=frozenset({'root vegetables', 'yogurt', 'nan',
'rolls/buns', 'tropical fruit'}), items_add=frozenset({'whole milk'}), con
fidence=0.8076923076923078, lift=3.161024212556246)])]
```

Question 6 - Advanced

Analyze the first relation obtained from the rules obtained previously & derive useful insights.

In [ ]:

```
print(result[0])
```

```
RelationRecord(items=frozenset({'baking powder', 'root vegetables', 'other
vegetables'}), support=0.002541942043721403, ordered_statistics=[OrderedSt
atistic(items_base=frozenset({'baking powder', 'root vegetables'}), items_
add=frozenset({'other vegetables'}), confidence=0.7142857142857143, lift=
3.691539674198634)])
```

From the above, you can observe the following for the first rule

- **Support =** 0.0025 implies the frequnecy of a transaction containing **root vegetables**, **other vegetables** and **baking powder**. Therefore, we can say that in every 10,000 transactions 25 transaction contains **root vegetables**, **other vegetables** and **baking powder**
- **Confidence =** 0.71 implies that if a customer buys both **root vegetables** and **baking powder**, we can say with 71% confidence that he/she will also buy **other vegetables**
- **Lift =** 3.69 implies that if a customer buys both **root vegetables** and **baking powder** he/she is 3.69 times more likely to buy **other vegetables**

Questions 7-12:

Scenario 3 - Movie Recommender System (All Bridging Questions)

Dataset Description

This dataset contains 5 attributes:

> **1. userId**: Represents ID of the user.
>
> **2. movieId**: Represents ID of the movie.
>
> **3. rating**: Represents rating of the movie made on a 5-star scale.
>
> **4. timestamp**: Represents seconds since midnight UTC of January 1, 1970.
>
> **5. original_title**: Represents seconds since midnight UTC of January 1, 1970.

Annie, a friend of yours, just watched a movie called **Alice in the Wonderland**. Predict movies similar to Alice in the Wonderland using a Recommender System written in Python

Tasks to be Performed

- Load and analyze the dataset and check for any null values present in the dataset - Beginner
- Count the ratings of the movies in the data set and also create a dataframe with the average ratings and the total number of ratings for each movie - Intermediate
- Sort the data frame according to the number of ratings received by each Movie - Intermediate
- Analyze the correlation that may exist between similar movies in the data set - Advanced
- Recommend to the user, movies that are similar to Alice in Wonderland - Intermediate
- Recommend to the user, movies that are similar to The Three Musketeers - Intermediate

Topics Covered

- Recommendation System

In [ ]:

```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/hyq3so7oc75ponm/movie_ratings.csv
```

```
--2020-07-16 09:15:11--  https://www.dropbox.com/s/hyq3so7oc75ponm/movie_r
atings.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:100:602
1:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... conne
cted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/hyq3so7oc75ponm/movie_ratings.csv [following]
--2020-07-16 09:15:11--  https://www.dropbox.com/s/raw/hyq3so7oc75ponm/mov
ie_ratings.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucf7198cf2461b4325c2df191957.dl.dropboxusercontent.com/c
d/0/inline/A7ni-_piu5jHO9lVwH5BereibYBG2T8cB5ZNXndV-a6c4ZSCj6UpdbnL0rcDi0s
t7ScicXKreAlRZPNy0dyu9M__nNeFRDWr8EYeOtGY4HlLHIRagKIJTef7g_1qhfarM_4/file#
[following]
--2020-07-16 09:15:11--  https://ucf7198cf2461b4325c2df191957.dl.dropboxus
ercontent.com/cd/0/inline/A7ni-_piu5jHO9lVwH5BereibYBG2T8cB5ZNXndV-a6c4ZSC
j6UpdbnL0rcDi0st7ScicXKreAlRZPNy0dyu9M__nNeFRDWr8EYeOtGY4HlLHIRagKIJTef7g_
1qhfarM_4/file
Resolving ucf7198cf2461b4325c2df191957.dl.dropboxusercontent.com (ucf7198c
f2461b4325c2df191957.dl.dropboxusercontent.com)... 162.125.65.15, 2620:10
0:6021:15::a27d:410f
Connecting to ucf7198cf2461b4325c2df191957.dl.dropboxusercontent.com (ucf7
198cf2461b4325c2df191957.dl.dropboxusercontent.com)|162.125.65.15|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 1870640 (1.8M) [text/plain]
Saving to: ‘movie_ratings.csv’

movie_ratings.csv   100%[===================>]   1.78M  --.-KB/s    in 0.0
3s

2020-07-16 09:15:12 (59.8 MB/s) - ‘movie_ratings.csv’ saved [1870640/18706
40]
```

Question 7 - Beginner (Bridging Question)

Load and analyze the dataset and also check for any null values that may be present in the dataset.

In [ ]:

```
import pandas as pd

df = pd.read_csv('/content/movie_ratings.csv')
df.head()
```

Out[ ]:

	userId	movieId	rating	timestamp	original_title
0	1	31	2.5	1260759144	Toy Story
1	1	1029	3.0	1260759179	Jumanji
2	1	1061	3.0	1260759182	Grumpier Old Men
3	1	1129	2.0	1260759185	Waiting to Exhale
4	1	1172	4.0	1260759205	Father of the Bride Part II

In [ ]:

```
df.shape
```

Out[ ]:

```
(45466, 5)
```

In [ ]:

```
df.describe()
```

Out[ ]:

	userId	movieId	rating	timestamp
count	45466.000000	45466.000000	45466.000000	4.546600e+04
mean	159.915145	12509.613601	3.553688	1.139278e+09
std	96.779400	26454.939394	1.052545	1.933925e+08
min	1.000000	1.000000	0.500000	8.282132e+08
25%	73.000000	999.000000	3.000000	9.650921e+08
50%	159.000000	2392.000000	4.000000	1.112545e+09
75%	243.000000	5418.000000	4.000000	1.304992e+09
max	324.000000	162376.000000	5.000000	1.476641e+09

In [ ]:

```
#Checking for Null Values
df.isnull().sum()
```

Out[ ]:

```
userId            0
movieId           0
rating            0
timestamp         0
original_title    0
dtype: int64
```

In [ ]:

```
df.nunique()
```

Out[ ]:

```
userId              324
movieId            6477
rating               10
timestamp         36392
original_title    43371
dtype: int64
```

Question 8 - Intermediate (Bridging Question)

Count the ratings of the movies in the dataset and also create a dataframe with the average ratings and the total number of scores for each movie.

In [ ]:

```
#Count the ratings of all movies
df.groupby('original_title')['rating'].count().sort_values(ascending=False).head()
```

Out[ ]:

```
original_title
Alice in Wonderland     8
Hamlet                  8
Cinderella              7
Les Misérables          7
The Three Musketeers    7
Name: rating, dtype: int64
```

In [ ]:

```
#Creating a dataframe with mean 'rating' and total number of ratings for each movie
ratings = pd.DataFrame(df.groupby('original_title')['rating'].mean())
ratings['Number of Ratings'] = pd.DataFrame(df.groupby('original_title')['rating'].coun
t())
ratings.head()
```

Out[ ]:

original_title	rating	Number of Ratings
!Women Art Revolution	4.5	1
#1 Cheerleader Camp	5.0	1
#Horror	2.5	1
#Pellichoopulu	4.0	1
#SELFIEPARTY	4.5	1

Question 9 - Intermediate (Bridging Question)

Sort the dataframe according to the number of ratings received by each movie.

In [ ]:

```
#Sorting values according to the 'num of rating column'
pivot_movie = df.pivot_table(index ='userId',columns ='original_title', values ='ratin
g')
pivot_movie.head()
```

Out[ ]:

original_title	!Women Art Revolution	#1 Cheerleader Camp	#Horror	#Pellichoopulu	#SELFIEPARTY	#chicagoGirl
userId						
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 43371 columns

In [ ]:

```
ratings.sort_values('Number of Ratings', ascending = False).head(5)
```

Out[ ]:

original_title	rating	Number of Ratings
Alice in Wonderland	3.875000	8
Hamlet	3.250000	8
The Three Musketeers	3.000000	7
Les Misérables	3.642857	7
A Christmas Carol	4.285714	7

Question 10 - Advanced (Bridging Question)

Analyze the correlation that may exist between similar movies in the dataset.

In [ ]:

```
#Analysing correlation with similar movies
alice_user_ratings = pivot_movie['Alice in Wonderland']
ttm_user_ratings = pivot_movie['The Three Musketeers']

alice_user_ratings.head()
```

Out[ ]:

```
userId
1   NaN
2   NaN
3   NaN
4   NaN
5   NaN
Name: Alice in Wonderland, dtype: float64
```

In [ ]:

```
#Analysing Correlation with Similar Movies
similar_to_alice = pivot_movie.corrwith(alice_user_ratings)#'corrwith' computes pairwis
e correlation between rows and columns of two dataframes
similar_to_ttm = pivot_movie.corrwith(ttm_user_ratings)

similar_to_alice.head()
```

Out[ ]:

```
original_title
!Women Art Revolution   NaN
#1 Cheerleader Camp     NaN
#Horror                 NaN
#Pellichoopulu          NaN
#SELFIEPARTY            NaN
dtype: float64
```

In [ ]:

```
corr_alice = pd.DataFrame(similar_to_alice, columns =['Correlation'])
corr_alice.dropna(inplace = True)
corr_alice.head()
```

Out[ ]:

original_title	Correlation
Adventures in Babysitting	1.0
Alice in Wonderland	1.0
Broken English	1.0
Countdown	1.0
Life	-1.0

In [ ]:

```
corr_alice = corr_alice.join(ratings['Number of Ratings'])
corr_alice.head()
```

Out[ ]:

original_title	Correlation	Number of Ratings
Adventures in Babysitting	1.0	2
Alice in Wonderland	1.0	8
Broken English	1.0	2
Countdown	1.0	2
Life	-1.0	4

Question 11 - Intermediate (Bridging Question)

Recommend to the user, movies that are similar to *Alice in Wonderland*.

In [ ]:

```
#Similar movies like Alice in Wonderland
corr_alice[corr_alice['Number of Ratings']>2].sort_values('Correlation', ascending = False).head(10)
```

Out[ ]:

original_title	Correlation	Number of Ratings
Alice in Wonderland	1.0	8
Life	-1.0	4
The Man in the Iron Mask	-1.0	3

Question 12 - Intermediate (Bridging Question)

Recommend to the user, movies that are similar to *The Three Musketeers*.

In [ ]:

```
#Movies similar to The Three Musketeers
corr_ttm = pd.DataFrame(similar_to_ttm, columns =['Correlation'])
corr_ttm.dropna(inplace = True)
corr_ttm = corr_ttm.join(ratings['Number of Ratings'])
corr_ttm.head()
```

Out[ ]:

original_title	Correlation	Number of Ratings
The Three Musketeers	1.0	7
Wuthering Heights	1.0	6

In [ ]:

```
corr_ttm[corr_ttm['Number of Ratings']>2].sort_values('Correlation', ascending = False).head(10)
```

Out[ ]:

original_title	Correlation	Number of Ratings
The Three Musketeers	1.0	7
Wuthering Heights	1.0	6

Questions 13-15:

Scenario 4 - Mining Association Rules on Wine Dataset

Dataset Description

The dataset contains 20 transactions with different kinds of alcohol being sold at the Liquor Shop over the course of a week.

Let us say that you are a new wine taster and want to use this dataset on alcohol to perform Market Basket Analysis using Apriori Algorithm on a dataset containing 20 transactions over a period of one week of a retail store to increase sales.

Tasks to be Performed

- Load, analyze, and pre-process the dataset and convert the dataframe into a list of lists - Beginner
- Implement the Apriori Algorithm from the Apyori package - Intermediate
- Analyze the first relation obtained from the rules obtained & derive useful insights - Advanced

Topics Covered

- Association Rule Mining
- Apriori Algorithm

In [ ]:

```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/4108bbm2mn536w3/Wine.xlsx
```

```
--2020-06-29 11:45:06--  https://www.dropbox.com/s/4108bbm2mn536w3/Wine.xl
sx
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:603
2:1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... conne
cted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/4108bbm2mn536w3/Wine.xlsx [following]
--2020-06-29 11:45:06--  https://www.dropbox.com/s/raw/4108bbm2mn536w3/Win
e.xlsx
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uce2a9a55a98167355736297803b.dl.dropboxusercontent.com/c
d/0/inline/A6k75F7N5FaVXmfSUmkS3HUx-fBJfZmDagSCoAYzj6-dajNDgfYMbBAXO9gsUgd
jZkpFFVzwVIppTR7C1_KXZct0kH_ZcrquxMsE8G1JhXpU2jDjhK3-CBbGiH2aZ_pEiQI/file#
[following]
--2020-06-29 11:45:06--  https://uce2a9a55a98167355736297803b.dl.dropboxus
ercontent.com/cd/0/inline/A6k75F7N5FaVXmfSUmkS3HUx-fBJfZmDagSCoAYzj6-dajND
gfYMbBAXO9gsUgdjZkpFFVzwVIppTR7C1_KXZct0kH_ZcrquxMsE8G1JhXpU2jDjhK3-CBbGiH
2aZ_pEiQI/file
Resolving uce2a9a55a98167355736297803b.dl.dropboxusercontent.com (uce2a9a5
5a98167355736297803b.dl.dropboxusercontent.com)... 162.125.82.15, 2620:10
0:6032:15::a27d:520f
Connecting to uce2a9a55a98167355736297803b.dl.dropboxusercontent.com (uce2
a9a55a98167355736297803b.dl.dropboxusercontent.com)|162.125.82.15|:443...
connected.
HTTP request sent, awaiting response... 302 Found
Location: /cd/0/inline2/A6kwV1CuXRmLwcpASYHdPCbZn1xr_gwIly6u-SPblWH40nnEKv
K3lnShuWB9Q49z9d0fNoQsNz5lizsIiOKxJKK4KCdODWzoroCJdmjFtVzjoIPc4l8_gmiq55d1
maOxqHP6m_shF8jNbu5s2BQ8mxQJU59R31MEFrbkIaHqWFezmxWKRkNalrjQCE2hzrxsSDZOvp
hsOM-gd4_e2oaePq4TcCKJaxwPvTePdcqChrLLGNRQmaDPBsmlkx06LGURGV229YtCrhiDg3ml
MiejMaGTRDTP1Kfpot9Bz9GCkn-_XLcXEEpAIKOPjY_RFH3UiWgnSjMVhrem0aIw2rL-DI4hg6
r4fe8EllQpqcv0M5FokQ/file [following]
--2020-06-29 11:45:07--  https://uce2a9a55a98167355736297803b.dl.dropboxus
ercontent.com/cd/0/inline2/A6kwV1CuXRmLwcpASYHdPCbZn1xr_gwIly6u-SPblWH40nn
EKvK3lnShuWB9Q49z9d0fNoQsNz5lizsIiOKxJKK4KCdODWzoroCJdmjFtVzjoIPc4l8_gmiq5
5d1maOxqHP6m_shF8jNbu5s2BQ8mxQJU59R31MEFrbkIaHqWFezmxWKRkNalrjQCE2hzrxsSDZ
OvphsOM-gd4_e2oaePq4TcCKJaxwPvTePdcqChrLLGNRQmaDPBsmlkx06LGURGV229YtCrhiDg
3mlMiejMaGTRDTP1Kfpot9Bz9GCkn-_XLcXEEpAIKOPjY_RFH3UiWgnSjMVhrem0aIw2rL-DI4
hg6r4fe8EllQpqcv0M5FokQ/file
Reusing existing connection to uce2a9a55a98167355736297803b.dl.dropboxuser
content.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 9035 (8.8K) [application/vnd.openxmlformats-officedocument.spreads
heetml.sheet]
Saving to: ‘Wine.xlsx’

Wine.xlsx           100%[===================>]   8.82K  --.-KB/s    in 0s

2020-06-29 11:45:07 (226 MB/s) - ‘Wine.xlsx’ saved [9035/9035]
```

Question 13 - Beginner

Load, analyze, and pre-process the dataset and convert the dataframe into a list of lists.

In [ ]:

```
import pandas as pd

df = pd.read_excel('/content/Wine.xlsx',header=None)
df.head()
```

Out[ ]:

	0	1	2	3	4
0	Royal Stag	Old Monk	Black Dog	NaN	Kingfisher
1	Old Monk	Royal Stag	NaN	Teachers	NaN
2	NaN	NaN	Kingfisher	NaN	NaN
3	Teachers	NaN	Royal Stag	Jack Daniels	NaN
4	Jack Daniels	Old Monk	NaN	NaN	Royal Stag

From the above, you can see that the dataset contains NaN values. That means that item was not purchased in that transaction.

In [ ]:

```
df.describe()
```

Out[ ]:

	0	1	2	3	4
count	13	11	11	11	7
unique	6	6	6	5	4
top	Royal Stag	Old Monk	Old Monk	Teachers	Kingfisher
freq	6	4	3	3	3

In [ ]:

```
df.shape
```

Out[ ]:

```
(20, 5)
```

Converting the dataframe into a list of lists

Apriori library requires the dataset to be in the form of a list of lists, where the entire dataset is a big list and each transaction in the dataset is stored as a list inside the bigger list.

In [ ]:

```
dataset=[]
for i in range(0, df.shape[0]):
    dataset.append([str(df.values[i,j]) for j in range(0, df.shape[1])])

dataset
```

Out[ ]:

```
[['Royal Stag', 'Old Monk', 'Black Dog', 'nan', 'Kingfisher'],
 ['Old Monk', 'Royal Stag', 'nan', 'Teachers', 'nan'],
 ['nan', 'nan', 'Kingfisher', 'nan', 'nan'],
 ['Teachers', 'nan', 'Royal Stag', 'Jack Daniels', 'nan'],
 ['Jack Daniels', 'Old Monk', 'nan', 'nan', 'Royal Stag'],
 ['nan', 'Royal Stag', 'nan', 'Jack Daniels', 'nan'],
 ['nan', 'nan', 'Old Monk', 'nan', 'nan'],
 ['Royal Stag', 'nan', 'Black Dog', 'nan', 'Johnny Walker'],
 ['nan', 'Jack Daniels', 'nan', 'Royal Stag', 'nan'],
 ['Royal Stag', 'nan', 'nan', 'Old Monk', 'nan'],
 ['nan', 'nan', 'Teachers', 'nan', 'Black Dog'],
 ['Black Dog', 'Teachers', 'nan', 'Jack Daniels', 'nan'],
 ['Royal Stag', 'nan', 'Teachers', 'Black Dog', 'Kingfisher'],
 ['Kingfisher', 'nan', 'Old Monk', 'nan', 'nan'],
 ['nan', 'Jack Daniels', 'nan', 'Teachers', 'Black Dog'],
 ['Teachers', 'Black Dog', 'Kingfisher', 'Old Monk', 'nan'],
 ['Black Dog', 'Old Monk', 'nan', 'nan', 'Kingfisher'],
 ['Royal Stag', 'nan', 'Jack Daniels', 'Old Monk', 'nan'],
 ['nan', 'Kingfisher', 'Old Monk', 'nan', 'nan'],
 ['Royal Stag', 'Old Monk', 'nan', 'Teachers', 'nan']]
```

Question 14 - Intermediate

Implement the Apriori Algorithm from the Apyori package.

In [ ]:

```
!pip install apyori
```

```
Requirement already satisfied: apyori in /usr/local/lib/python3.6/dist-pac
kages (1.1.2)
```

In [ ]:

```
import pandas as pd
import numpy as np
from apyori import apriori
import warnings
warnings.filterwarnings("ignore")

rules = apriori(dataset, min_support=0.002,min_confidence=0.7,min_lift=3,min_length=1,
use_colnames=True)
result = list(rules)

result
```

Out[ ]:

```
[RelationRecord(items=frozenset({'Johnny Walker', 'Black Dog', 'Royal Sta
g'}), support=0.05, ordered_statistics=[OrderedStatistic(items_base=frozen
set({'Johnny Walker'}), items_add=frozenset({'Royal Stag', 'Black Dog'}),
confidence=1.0, lift=6.666666666666667)]),
 RelationRecord(items=frozenset({'Johnny Walker', 'nan', 'Black Dog', 'Roy
al Stag'}), support=0.05, ordered_statistics=[OrderedStatistic(items_base=
frozenset({'Johnny Walker'}), items_add=frozenset({'Black Dog', 'nan', 'Ro
yal Stag'}), confidence=1.0, lift=6.666666666666667), OrderedStatistic(ite
ms_base=frozenset({'Johnny Walker', 'nan'}), items_add=frozenset({'Royal S
tag', 'Black Dog'}), confidence=1.0, lift=6.666666666666667)])]
```

Question 15 - Advanced

Analyze the first relation obtained from the rules above & derive useful insights.

In [ ]:

```
result[0]
```

Out[ ]:

```
RelationRecord(items=frozenset({'Johnny Walker', 'Black Dog', 'Royal Sta
g'}), support=0.05, ordered_statistics=[OrderedStatistic(items_base=frozen
set({'Johnny Walker'}), items_add=frozenset({'Royal Stag', 'Black Dog'}),
confidence=1.0, lift=6.666666666666667)])
```

From the above, you can observe the following for the first rule

- **Confidence =** 1.0 implies that if a customer buys both **Johnny Walker**, we can say with 100% confidence that he/she will also buy **Royal Stag** and **Black Dog**
- **Lift =** 6.67 implies that if a customer buys both **Johnny Walker** he/she is 6.67 times more likely to buy **Royal Stag** and **Black Dog**

Module 4 - Time Series Analysis

- Time Series is a set of observations taken at specified times, usually at equal intervals.
- It is used to predict future values based on the previously observed values

Components of Time Series

- **Trend** - Variation in data over a long period of time
- **Seasonality** - Variation in data occurring at regular intervals of time
- **Cyclical Patterns** - Data pattern with uncertain timings and fluctuations
- **Irregularity** - Data with short-periodic, non-repeated and unpredictable events

Questions 1-4:

Scenario 1 - Shampoo Sales Analysis

Garnier is a mass-market cosmetics brand of French cosmetics company L'Oréal, a french personal care company headquartered in Clichy, Hauts-de-Seine with a registered office in Paris. It is the world's largest cosmetics company. They have collected sales data of their shampoo for 3 years.

In Time Series Forecasting, we use a model to predict future values based on previously observed values. It is a useful method because L'Oréal can use it to predict/forecast future sales of their shampoo.

Problem Statement

A Cosmetic Company called **L'Oréal** has the data of its shampoo sales for a period of 3 years. The data is classified in date/time and the sales data of shampoo.

So, being a Data Scientist, you must -

Perform Time Series Analysis on the given dataset.

Dataset Description

The dataset describes the monthly number of sales of shampoo over a 3 year period.

The units are a sales count, and there are 36 observations. The original dataset is credited to Makridakis, Wheelwright and Hyndman (1998).

Here's a brief description of the dataset:

- **Month** - Month
- **Sales** - Sale of Shampoo

Tasks to be Performed

- Load, analyze, pre-process and visualize the dataset to determine the trend - Beginner
- Check the Stationarity of the dataset using the Rolling Statistics technique and also plot the Rolling Statistics - Intermediate
- Check the Stationarity of the dataset using the Dickey-Fuller Test - Intermediate
- Convert Non-Stationary data to Stationary data using the **Log** method and then, check for Stationarity using both the Rolling Statistics and Dickey-Fuller Test - Intermediate

Topics Covered

- Rolling Statistics
- Dickey-Fuller Test

In [ ]:

```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/pljx6u7zpz9dmqp/shampoo.csv
```

```
--2020-07-16 09:20:28--  https://www.dropbox.com/s/pljx6u7zpz9dmqp/shampo
o.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.1.1, 2620:100:6016:
1::a27d:101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.1.1|:443... connec
ted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/pljx6u7zpz9dmqp/shampoo.csv [following]
--2020-07-16 09:20:28--  https://www.dropbox.com/s/raw/pljx6u7zpz9dmqp/sha
mpoo.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucfefe99f7a5818278d20d206b28.dl.dropboxusercontent.com/c
d/0/inline/A7mYDmh3WviNaIaLP7gMLsc7CBLht7z2FjWMADzB6IbqK6rm9mGauThzvIK9TKQ
8xFyKCaBKENy7ksMt-RNZkb4W5Qfi8WSwgrcB6DP2DLpk2N29x43bDuKOInNTdJmL-UM/file#
[following]
--2020-07-16 09:20:28--  https://ucfefe99f7a5818278d20d206b28.dl.dropboxus
ercontent.com/cd/0/inline/A7mYDmh3WviNaIaLP7gMLsc7CBLht7z2FjWMADzB6IbqK6rm
9mGauThzvIK9TKQ8xFyKCaBKENy7ksMt-RNZkb4W5Qfi8WSwgrcB6DP2DLpk2N29x43bDuKOIn
NTdJmL-UM/file
Resolving ucfefe99f7a5818278d20d206b28.dl.dropboxusercontent.com (ucfefe99
f7a5818278d20d206b28.dl.dropboxusercontent.com)... 162.125.1.15, 2620:100:
6016:15::a27d:10f
Connecting to ucfefe99f7a5818278d20d206b28.dl.dropboxusercontent.com (ucfe
fe99f7a5818278d20d206b28.dl.dropboxusercontent.com)|162.125.1.15|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 519 [text/plain]
Saving to: ‘shampoo.csv’

shampoo.csv         100%[===================>]     519  --.-KB/s    in 0s

2020-07-16 09:20:29 (52.6 MB/s) - ‘shampoo.csv’ saved [519/519]
```

Question 1 - Beginner

Load, analyze, pre-process and visualize the dataset to determine the trend.

In [ ]:

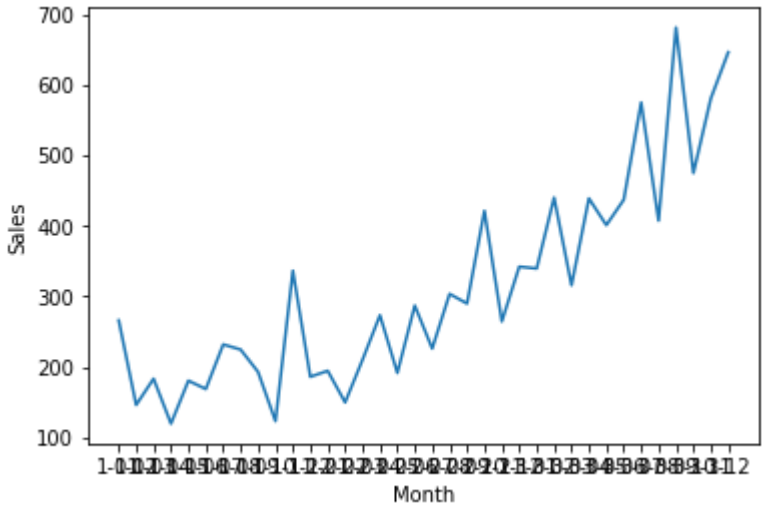
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('/content/shampoo.csv',parse_dates=True, index_col='Month')
df.head()
```

Out[ ]:

Month	Sales
1-01	266.0
1-02	145.9
1-03	183.1
1-04	119.3
1-05	180.3

In [ ]:

```
plt.xlabel('Month')
plt.ylabel('Sales')
plt.plot(df)
plt.show()
```



From the above, you can see that the data has an overall increasing trend.

Question 2 - Intermediate

Check the stationarity of the dataset using the Rolling Statistics technique and also plot the Rolling Statistics.

In [ ]:

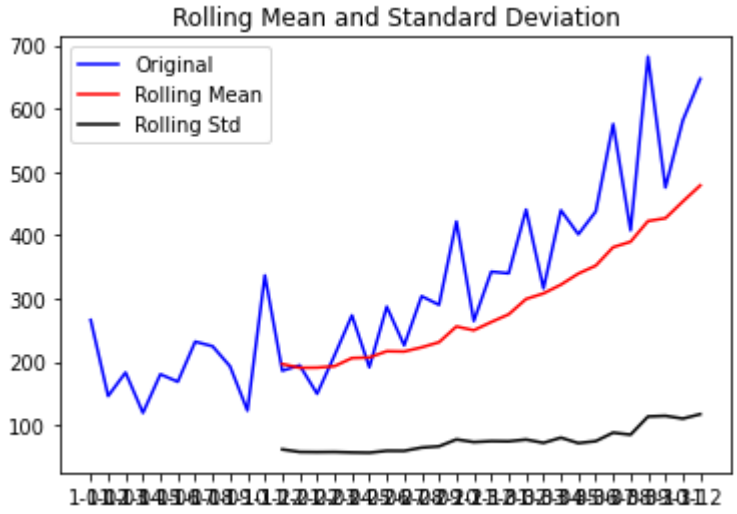
```
#Determine Rolling Statistics
rollmean = df.rolling(window = 12).mean() #Window of 12 Months
rollstd = df.rolling(window = 12).std()

print(rollmean, rollstd)
```

```
            Sales
Month
1-01          NaN
1-02          NaN
1-03          NaN
1-04          NaN
1-05          NaN
1-06          NaN
1-07          NaN
1-08          NaN
1-09          NaN
1-10          NaN
1-11          NaN
1-12   196.458333
2-01   190.483333
2-02   190.783333
2-03   193.033333
2-04   205.866667
2-05   206.791667
2-06   216.666667
2-07   216.183333
2-08   222.775000
2-09   230.866667
2-10   255.758333
2-11   249.758333
2-12   262.791667
3-01   274.908333
3-02   299.150000
3-03   307.966667
3-04   321.800000
3-05   339.291667
3-06   351.825000
3-07   380.950000
3-08   389.616667
3-09   422.291667
3-10   426.766667
3-11   453.166667
3-12   478.550000            Sales
Month
1-01          NaN
1-02          NaN
1-03          NaN
1-04          NaN
1-05          NaN
1-06          NaN
1-07          NaN
1-08          NaN
1-09          NaN
1-10          NaN
1-11          NaN
1-12    61.606913
2-01    57.595562
2-02    57.351085
2-03    57.551532
2-04    56.780091
2-05    56.414963
2-06    59.395490
2-07    59.284642
2-08    64.464552
2-09    66.424274
2-10    77.354726
```

```
2-11    73.203632
2-12    74.708007
3-01    74.379964
3-02    77.144793
3-03    71.910529
3-04    80.132107
3-05    71.527038
3-06    74.640412
3-07    88.061182
3-08    84.814427
3-09   113.562594
3-10   114.586286
3-11   110.213325
3-12   117.212383
```

In [ ]:

```
#Plot Rolling Statistics
orig = plt.plot(df, color = 'blue', label = 'Original')
mean = plt.plot(rollmean, color = 'red', label = 'Rolling Mean')
std = plt.plot(rollstd, color = 'black', label = 'Rolling Std')

plt.legend(loc = 'best')
plt.title("Rolling Mean and Standard Deviation")
plt.show()
```



From the above, you can see that both the **Mean** and **Standard Deviation** are not constant. So, our data is not stationary.

Question 3 - Intermediate

Check the Stationarity of the dataset using the Dickey-Fuller Test.

Null Hypothesis - Data is not Stationary

Alternate Hypothesis - Data is Stationary

In [ ]:

```
from statsmodels.tsa.stattools import adfuller

print("Results of Dickey Fuller Test:")

dftest = adfuller(df['Sales'], autolag = 'AIC') #AIC is Akaike Information Criterion
dfoutput = pd.Series(dftest[0:4], index = ['Test Statistic', 'p-value', '#Lags Used',
'No. of Observations Used'])
for key, value in dftest[4].items():
  dfoutput['Critical Value: %s'%key] = value

print(dfoutput)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: F
utureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm

Results of Dickey Fuller Test:
Test Statistic                3.060142
p-value                       1.000000
#Lags Used                   10.000000
No. of Observations Used     25.000000
Critical Value: 1%           -3.723863
Critical Value: 5%           -2.986489
Critical Value: 10%          -2.632800
dtype: float64
```

From above, you can see that,

- P-value should always be less but, here we have a very large p-value
- Critical Value should be more than the Test Statistic

So, here we **cannot reject the Null Hypothesis**, and we can say that the **Data is not Stationary**.

Question 4 - Intermediate

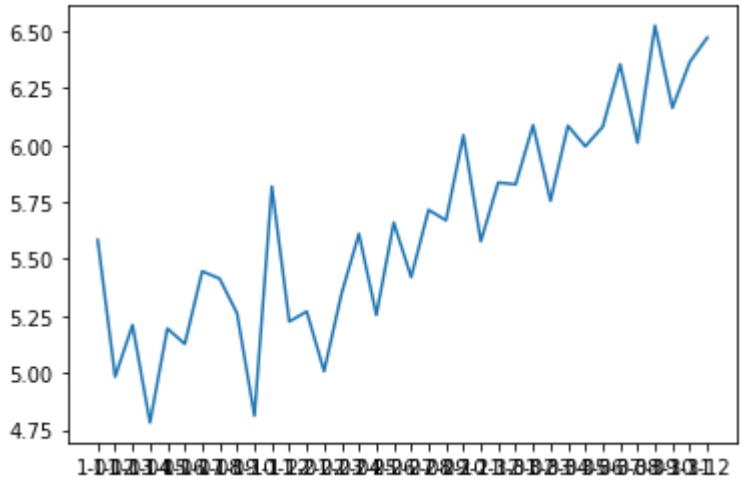
Convert Non-Stationary data to Stationary data using the **Log** method, and then check for Stationarity using both the Rolling Statistics and Dickey-Fuller Test.

In [ ]:

```
import numpy as np

df_log = np.log(df) #log of the dataset
plt.plot(df_log)
```

Out[ ]:

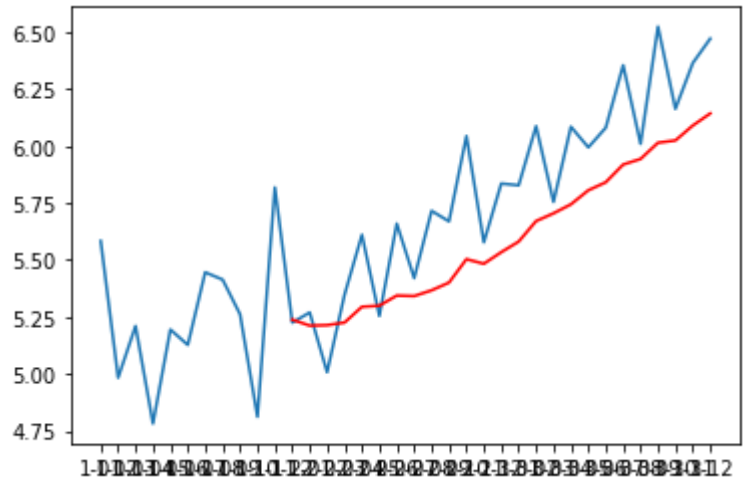
```
[<matplotlib.lines.Line2D at 0x7f9ef90c06a0>]
```



From the above, you can see that the numbers on y-axis have changed because the scale itself has changed.

In [ ]:

```
ma = df_log.rolling(window = 12).mean() #Window of 12 Months
mstd = df_log.rolling(window = 12).std()

plt.plot(df_log)
plt.plot(ma, color = 'Red')
plt.show()
```



In [ ]:

```
df_diff = df_log - ma #Difference between the Moving Average and the Actual Number of P
asssengers

#df_diff.head(12)

df_diff.dropna(inplace = True)
df_diff.head(12)
```

Out[ ]:

Month	Sales
1-12	-0.012810
2-01	0.057558
2-02	-0.206580
2-03	0.122245
2-04	0.316154
2-05	-0.045029
2-06	0.315709
2-07	0.078873
2-08	0.348896
2-09	0.268731
2-10	0.540528
2-11	0.094375

In [ ]:

```
def test_stationarity(timeseries):

    #Determing rolling statistics
    ma = timeseries.rolling(window=12).mean()
    mstd = timeseries.rolling(window=12).std()

    #Plot rolling statistics:
    plt.figure(figsize=(12,3))
    plt.grid(True)
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(ma, color='red', label='Rolling Mean')
    std = plt.plot(mstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print("Results of Dickey Fuller Test:")

    dftest = adfuller(timeseries['Sales'], autolag = 'AIC') #AIC is Akaike Information
 Criterion

    dfoutput = pd.Series(dftest[0:4], index = ['Test Statistic', 'p-value', '#Lags Use
d', 'No. of Observations Used'])

    for key, value in dftest[4].items():
      dfoutput['Critical Value: %s'%key] = value

    print(dfoutput)
```

Questions 5-10:

Scenario 2 - Forecast Star Air Passenger Traffic (All Bridging Questions)

Star Air is a British Airline with its head office in Crawley, England, with some basic data of its passengers across months which is classified in date and the number of passengers traveling per month.

In Time Series Forecasting, we use a model to predict future values based on previously observed values. It is a useful method because Star Air can forecast future passenger traffic and plan flight routes accordingly.

Problem Statement

An Airline called **Star Air** has the data of its passengers across months. The data is classified in date/time and the passengers traveling per month.

So, being an Analyst, you must -

Build a model to forecast the demand (passenger traffic) in Airplanes in the future in Python.

Dataset Description

Here's a brief description of the dataset:

- **Month** - Dates from 1949 till 1960 monthwise
- **#Passengers** - Number of Passengers traveling per month

Tasks to be Performed

- Load, analyze, pre-process and visualize the dataset to determine the trend - Beginner
- Check the Stationarity of the dataset using the Rolling Statistics technique and also plot the Rolling Statistics - Intermediate
- Check the Stationarity of the dataset using the Dickey-Fuller Test - Intermediate
- Convert Non-Stationary data to Stationary data using the **Log** method and then, check for Stationarity using both the Rolling Statistics and Dickey-Fuller Test - Intermediate
- Plot the Auto-Correlation and Partial Auto-Correlation Graph - Intermediate
- Build the ARIMA Model and forecast the demand - Advanced

Topics Covered

- Rolling Statistics
- Dickey-Fuller Test
- Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF)
- ARIMA Model

In [ ]:

```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/ogl8t4g1wfn8duh/AirPassengers.csv
```

```
--2020-06-29 11:33:25--  https://www.dropbox.com/s/ogl8t4g1wfn8duh/AirPass
engers.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:
1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connec
ted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/ogl8t4g1wfn8duh/AirPassengers.csv [following]
--2020-06-29 11:33:25--  https://www.dropbox.com/s/raw/ogl8t4g1wfn8duh/Air
Passengers.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uccf1167b513e2faf5d107a1786e.dl.dropboxusercontent.com/c
d/0/inline/A6mwHCOXzEtdZtpmYUb95EbkVsqb_FgLqZiJY16JyirsEHDGS3vckycHfMLFMfp
pRVAFn3mHUBK8g4nNww4ZsgkiMkwln-9hcM--K2KxNbePLFruQt3tANWCQaO6C5E-bNo/file#
[following]
--2020-06-29 11:33:26--  https://uccf1167b513e2faf5d107a1786e.dl.dropboxus
ercontent.com/cd/0/inline/A6mwHCOXzEtdZtpmYUb95EbkVsqb_FgLqZiJY16JyirsEHDG
S3vckycHfMLFMfppRVAFn3mHUBK8g4nNww4ZsgkiMkwln-9hcM--K2KxNbePLFruQt3tANWCQa
O6C5E-bNo/file
Resolving uccf1167b513e2faf5d107a1786e.dl.dropboxusercontent.com (uccf1167
b513e2faf5d107a1786e.dl.dropboxusercontent.com)... 162.125.5.15, 2620:100:
601d:15::a27d:50f
Connecting to uccf1167b513e2faf5d107a1786e.dl.dropboxusercontent.com (uccf
1167b513e2faf5d107a1786e.dl.dropboxusercontent.com)|162.125.5.15|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 1746 (1.7K) [text/plain]
Saving to: ‘AirPassengers.csv’

AirPassengers.csv   100%[===================>]   1.71K  --.-KB/s    in 0s

2020-06-29 11:33:26 (289 MB/s) - ‘AirPassengers.csv’ saved [1746/1746]
```

Question 5 - Beginner (Bridging Question)

Load, analyze, pre-process and visualize the dataset to determine the trend.

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df=pd.read_csv('/content/AirPassengers.csv', parse_dates=True, index_col='Month')
df.head()
```

Out[ ]:

Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

In [ ]:

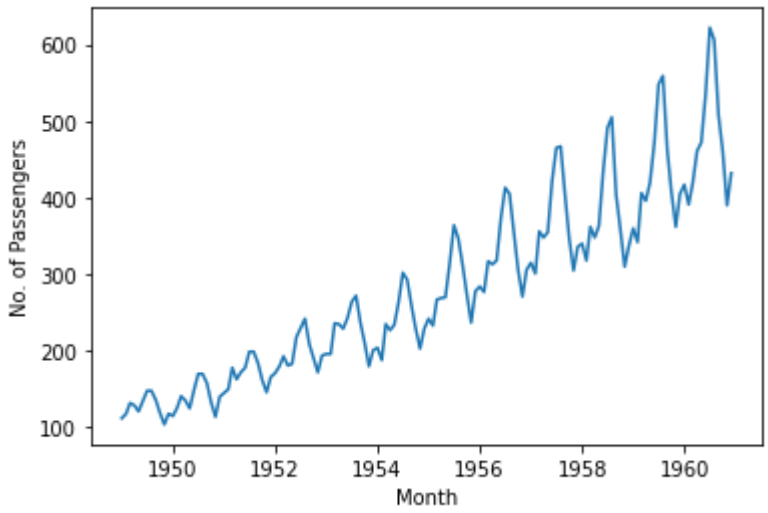
```
df.shape
```

Out[ ]:

```
(144, 1)
```

In [ ]:

```
plt.xlabel('Month')
plt.ylabel('No. of Passengers')
plt.plot(df)
plt.show()
```



From the above, you can see that there is an **overall increasing trend** in the dataset with some variations.

Question 6 - Intermediate (Bridging Question)

Check the Stationarity of the dataset using the Rolling Statistics technique and also plot the Rolling Statistics.

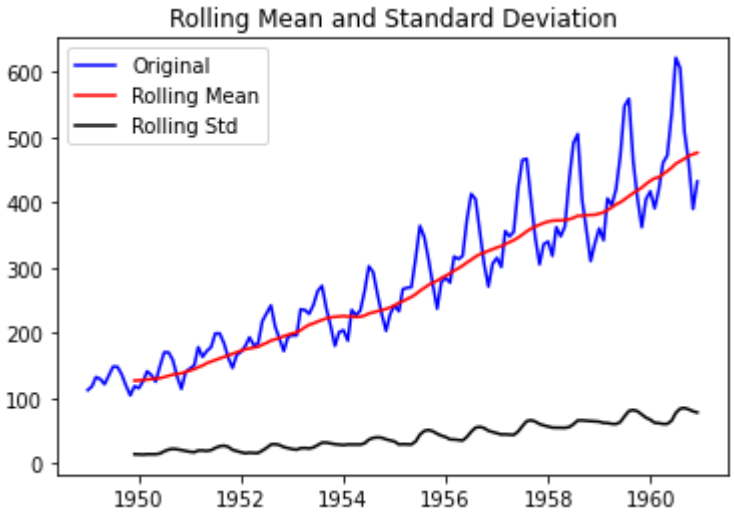
In [ ]:

```
#Determine Rolling Statistics
rollmean = df.rolling(window = 12).mean() #Window of 12 Months
rollstd = df.rolling(window = 12).std()

print(rollmean, rollstd)
```

```
            #Passengers
Month
1949-01-01          NaN
1949-02-01          NaN
1949-03-01          NaN
1949-04-01          NaN
1949-05-01          NaN
...                 ...
1960-08-01   463.333333
1960-09-01   467.083333
1960-10-01   471.583333
1960-11-01   473.916667
1960-12-01   476.166667

[144 rows x 1 columns]             #Passengers
Month
1949-01-01          NaN
1949-02-01          NaN
1949-03-01          NaN
1949-04-01          NaN
1949-05-01          NaN
...                 ...
1960-08-01    83.630500
1960-09-01    84.617276
1960-10-01    82.541954
1960-11-01    79.502382
1960-12-01    77.737125

[144 rows x 1 columns]
```

In [ ]:

```
#Plot Rolling Statistics
orig = plt.plot(df, color = 'blue', label = 'Original')
mean = plt.plot(rollmean, color = 'red', label = 'Rolling Mean')
std = plt.plot(rollstd, color = 'black', label = 'Rolling Std')

plt.legend(loc = 'best')
plt.title("Rolling Mean and Standard Deviation")
plt.show()
```



From the above, you can see that both the **Mean** and **Standard Deviation** are not constant. So, our data is not stationary.

Question 7 - Intermediate (Bridging Question)

Check the Stationarity of the dataset using the Dickey-Fuller Test.

Null Hypothesis - Data is not Stationary

Alternate Hypothesis - Data is Stationary

In [ ]:

```
from statsmodels.tsa.stattools import adfuller

print("Results of Dickey Fuller Test:")

dftest = adfuller(df['#Passengers'], autolag = 'AIC') #AIC is Akaike Information Criter
ion
dfoutput = pd.Series(dftest[0:4], index = ['Test Statistic', 'p-value', '#Lags Used',
'No. of Observations Used'])
for key, value in dftest[4].items():
  dfoutput['Critical Value: %s'%key] = value

print(dfoutput)
```

```
Results of Dickey Fuller Test:
Test Statistic                  0.815369
p-value                         0.991880
#Lags Used                     13.000000
No. of Observations Used      130.000000
Critical Value: 1%             -3.481682
Critical Value: 5%             -2.884042
Critical Value: 10%            -2.578770
dtype: float64
```

From the above, you can see that,

- P-value should always be less but, here we have a very large p-value, i.e., 0.9
- Critical Value should be more than the Test Statistic

So, here we **cannot reject the Null Hypothesis**, and we can say that the **Data is not Stationary**.

Question 8 - Intermediate (Bridging Question)

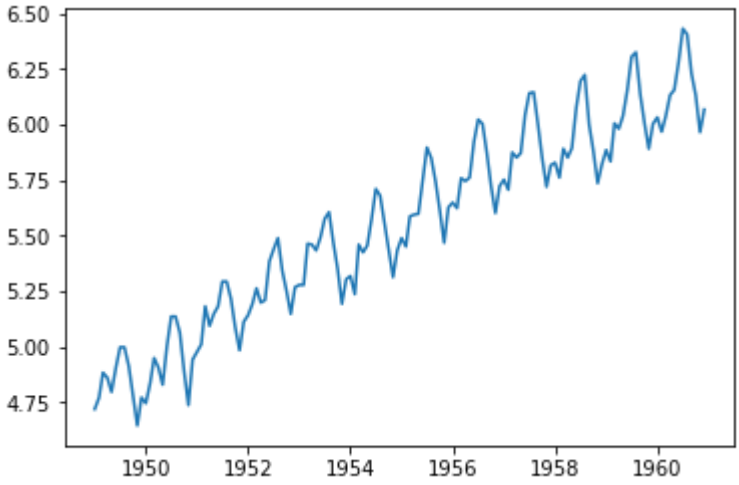
Convert Non-Stationary data to Stationary data using the **Log** method, and then check for Stationarity using both the Rolling Statistics and Dickey-Fuller Test.

In [ ]:

```
df_log = np.log(df) #Log of the dataset
plt.plot(df_log)
```

Out[ ]:

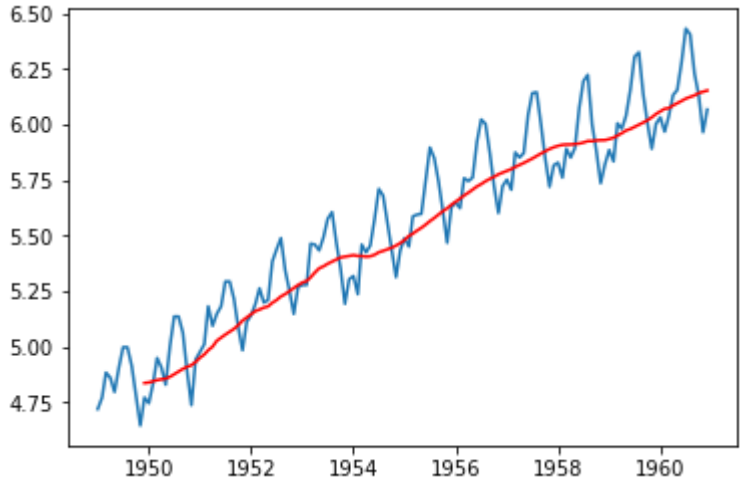
```
[<matplotlib.lines.Line2D at 0x7f9ef7c62b70>]
```



From the above, you can see that the numbers on y-axis have changed because the scale itself has changed.

In [ ]:

```
ma = df_log.rolling(window = 12).mean() #Window of 12 Months
mstd = df_log.rolling(window = 12).std()

plt.plot(df_log)
plt.plot(ma, color = 'Red')
plt.show()
```



From the above, you can see that the Mean is not Stationary but it is quite better than the previous one.

In [ ]:

```
df_diff = df_log - ma #Difference between the Moving Average and the Actual Number of P
asssengers

#df_diff.head(12)

df_diff.dropna(inplace = True)
df_diff.head(12)
```

Out[ ]:

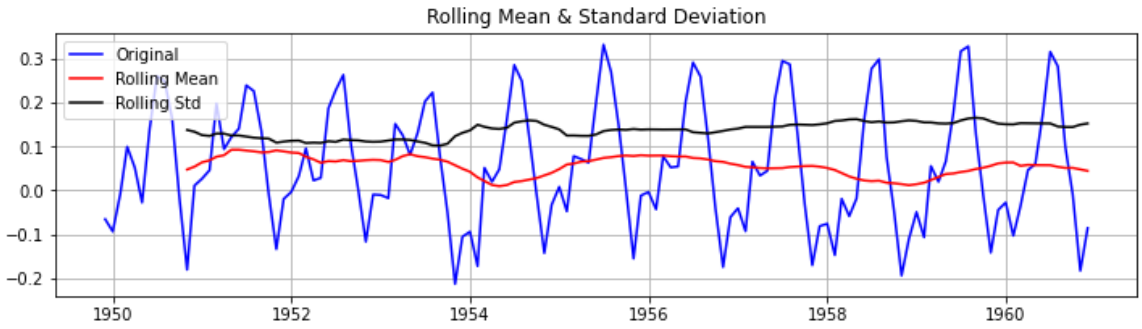
Month	#Passengers
1949-12-01	-0.065494
1950-01-01	-0.093449
1950-02-01	-0.007566
1950-03-01	0.099416
1950-04-01	0.052142
1950-05-01	-0.027529
1950-06-01	0.139881
1950-07-01	0.260184
1950-08-01	0.248635
1950-09-01	0.162937
1950-10-01	-0.018578
1950-11-01	-0.180379

In [ ]:

```
def test_stationarity(timeseries):

    #Determing rolling statistics
    ma = timeseries.rolling(window=12).mean()
    mstd = timeseries.rolling(window=12).std()

    #Plot rolling statistics:
    plt.figure(figsize=(12,3))
    plt.grid(True)
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(ma, color='red', label='Rolling Mean')
    std = plt.plot(mstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print("Results of Dickey Fuller Test:")

    dftest = adfuller(timeseries['#Passengers'], autolag = 'AIC') #AIC is Akaike Inform
ation Criterion

    dfoutput = pd.Series(dftest[0:4], index = ['Test Statistic', 'p-value', '#Lags Use
d', 'No. of Observations Used'])

    for key, value in dftest[4].items():
      dfoutput['Critical Value: %s'%key] = value

    print(dfoutput)
```

In [ ]:

```
test_stationarity(df_diff)
```




```
Results of Dickey Fuller Test:
Test Statistic                -3.162908
p-value                        0.022235
#Lags Used                    13.000000
No. of Observations Used     119.000000
Critical Value: 1%            -3.486535
Critical Value: 5%            -2.886151
Critical Value: 10%           -2.579896
dtype: float64
```

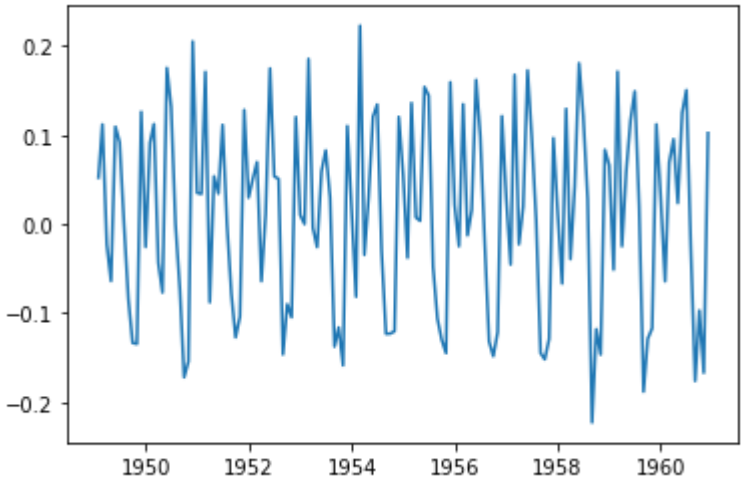
From the above, you can see that the **P-value** is relatively less and the Critical Value and Test Statistic value is almost equal which helps us determine whether the data is stationary or not.

Question 9 - Intermediate (Bridging Question)

Plot the Auto-Correlation and Partial Auto-Correlation Graph.

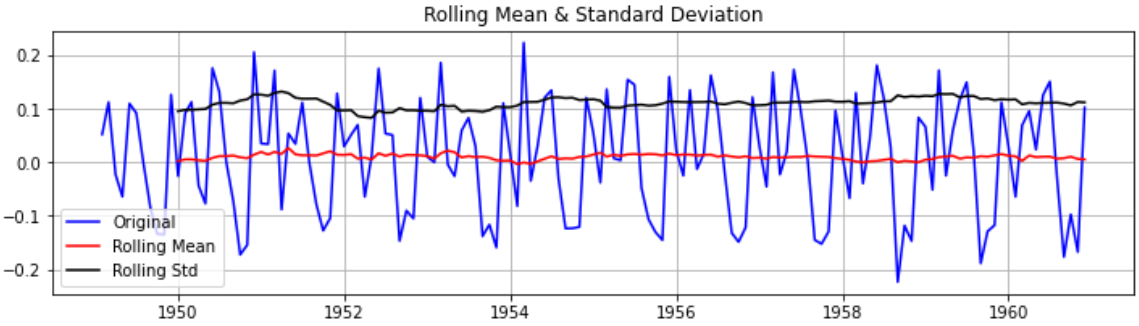
In [ ]:

```
df_log_shift = df_log - df_log.shift()

plt.plot(df_log_shift)
plt.show()
```



In [ ]:

```
df_log_shift.dropna(inplace = True)
test_stationarity(df_log_shift)
```



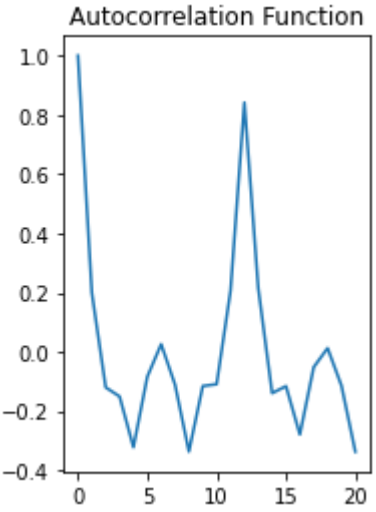

```
Results of Dickey Fuller Test:
Test Statistic                -2.717131
p-value                        0.071121
#Lags Used                    14.000000
No. of Observations Used     128.000000
Critical Value: 1%            -3.482501
Critical Value: 5%            -2.884398
Critical Value: 10%           -2.578960
dtype: float64
```

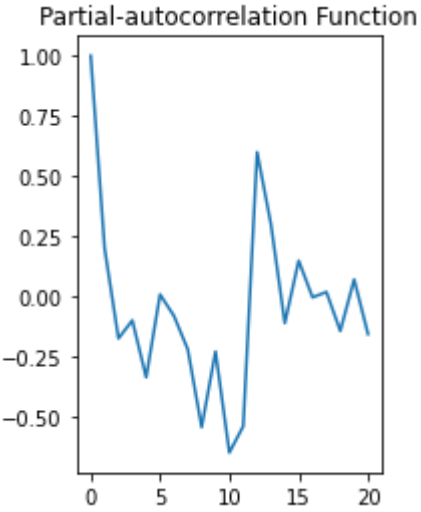
From the above graph, you can see that the time series is **Stationary**.

In [ ]:

```
from statsmodels.tsa.stattools import acf, pacf

lag_acf = acf(df_log_shift, nlags = 20)
lag_pacf = pacf(df_log_shift, nlags = 20, method = 'ols')

#Plot ACF
plt.subplot(121)
plt.plot(lag_acf)
plt.title("Autocorrelation Function")
plt.show()

#Plot PACF
plt.subplot(122)
plt.plot(lag_pacf)
plt.title("Partial-autocorrelation Function")
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/stattools.py:541: F
utureWarning: fft=True will become the default in a future version of stat
smodels. To suppress this warning, explicitly set fft=False.
  warnings.warn(msg, FutureWarning)
```





Now, in order to calculate the **p-value** and **q-value**, you need to check the value where the graph drops to 0 for the first time. Therefore, p = 2 & q = 2.

Question 10 - Advanced (Bridging Question)

Build the ARIMA Model and forecast the demand.

In [ ]:

```
from statsmodels.tsa.arima_model import ARIMA

#AR Model
model = ARIMA(df_log, order = (2, 1, 2))
results_AR = model.fit()

results_AR.summary()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
```

Out[ ]:

ARIMA Model Results

Dep. Variable:	D.#Passengers	**No. Observations:**	143
Model:	ARIMA(2, 1, 2)	**Log Likelihood**	149.640
Method:	css-mle	**S.D. of innovations**	0.084
Date:	Mon, 29 Jun 2020	**AIC**	-287.281
Time:	12:57:30	**BIC**	-269.504
Sample:	02-01-1949	**HQIC**	-280.057
	- 12-01-1960		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0096	0.003	3.697	0.000	0.005	0.015
ar.L1.D.#Passengers	1.6293	0.039	41.868	0.000	1.553	1.706
ar.L2.D.#Passengers	-0.8946	0.039	-23.127	0.000	-0.970	-0.819
ma.L1.D.#Passengers	-1.8270	0.036	-51.303	0.000	-1.897	-1.757
ma.L2.D.#Passengers	0.9245	0.036	25.568	0.000	0.854	0.995

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.9106	-0.5372j	1.0573	-0.0848
AR.2	0.9106	+0.5372j	1.0573	0.0848
MA.1	0.9881	-0.3245j	1.0400	-0.0505
MA.2	0.9881	+0.3245j	1.0400	0.0505

In [ ]:

```
#Analyse residual errors. They should be normal
residuals = pd.DataFrame(results_AR.resid)
print(residuals.describe())
```

```
                0
count  143.000000
mean     0.001691
std      0.085116
min     -0.193387
25%     -0.063327
50%     -0.005020
75%      0.074605
max      0.210671
```

In [ ]:

```
results_AR.forecast(steps=120)
```

Out[ ]:

```
(array([6.09553389, 6.15281399, 6.22442963, 6.29241102, 6.3416472 ,
        6.36359368, 6.35784693, 6.3313931 , 6.29597544, 6.26447715,
        6.24738324, 6.25025169, 6.27275833, 6.3094031 , 6.35151493,
        6.38988657, 6.41727384, 6.43011056, 6.4290669 , 6.41842488,
        6.40456152, 6.39403611, 6.39183077, 6.40019497, 6.41833688,
        6.44295405, 6.46937434, 6.49293985, 6.51024118, 6.51989032,
        6.52267575, 6.52112382, 6.5186452 , 6.51853681, 6.52311914,
        6.53322353, 6.54812864, 6.56591553, 6.58410305, 6.60036536,
        6.61313262, 6.62192775, 6.62737781, 6.6309312 , 6.63438679,
        6.63937975, 6.64696495, 6.65739832, 6.67015317, 6.68414244,
        6.69806619, 6.71077894, 6.72157724, 6.73033975, 6.737498  ,
        6.74386367, 6.75037311, 6.7578258 , 6.7666867 , 6.77699812,
        6.78841309, 6.80032843, 6.81207184, 6.82308749, 6.83307124,
        6.8420248 , 6.85022299, 6.85811207, 6.86617326, 6.87479137,
        6.88416288, 6.8942637 , 6.90487876, 6.91567925, 6.92632183,
        6.93654126, 6.94621252, 6.95536922, 6.96417792, 6.97287997,
        6.98171954, 6.99087858, 7.00043512, 7.01035347, 7.02050574,
        7.03071545, 7.0408095 , 7.05066371, 7.06023064, 7.06954406,
        7.07870143, 7.08783136, 7.09705615, 7.10646005, 7.11607092,
        7.12585876, 7.13574977, 7.14565059, 7.15547508, 7.16516643,
        7.17470917, 7.18412884, 7.19348099, 7.2028332 , 7.2122459 ,
        7.22175712, 7.23137473, 7.24107755, 7.25082403, 7.26056541,
        7.27025942, 7.27988083, 7.28942632, 7.29891307, 7.30837201,
        7.31783822, 7.32734114, 7.33689735, 7.34650756, 7.35615808]),
 array([0.08384711, 0.10749461, 0.11568694, 0.11702774, 0.11703496,
        0.11744017, 0.11762249, 0.11778712, 0.12024162, 0.12736043,
        0.13870961, 0.15118794, 0.16157815, 0.16834396, 0.1717733 ,
        0.17311982, 0.17358732, 0.17385459, 0.17430217, 0.17543332,
        0.17788148, 0.18195723, 0.18726219, 0.19283395, 0.19769247,
        0.20130641, 0.20369057, 0.20519834, 0.2062527 , 0.20721   ,
        0.20836972, 0.21000328, 0.21229746, 0.21524481, 0.21860182,
        0.22198259, 0.22503067, 0.22755298, 0.22954809, 0.231148  ,
        0.23253828, 0.23390536, 0.23541312, 0.23718314, 0.23926541,
        0.24161541, 0.24410374, 0.2465636 , 0.24885259, 0.25089632,
        0.25269736, 0.25431631, 0.25584315, 0.25737231, 0.25898393,
        0.26072853, 0.26261551, 0.26461092, 0.26664901, 0.26865484,
        0.27056852, 0.272361  , 0.27403704, 0.27562777, 0.27717832,
        0.2787353 , 0.28033594, 0.28200021, 0.28372686, 0.28549513,
        0.28727214, 0.28902349, 0.2907234 , 0.29236086, 0.29394044,
        0.29547879, 0.29699872, 0.29852279, 0.30006784, 0.30164131,
        0.30324017, 0.30485271, 0.30646259, 0.30805388, 0.30961544,
        0.31114341, 0.31264127, 0.314118  , 0.31558507, 0.3170533 ,
        0.31853011, 0.32001805, 0.32151461, 0.32301341, 0.32450639,
        0.32598616, 0.32744798, 0.32889071, 0.33031669, 0.33173076,
        0.33313865, 0.33454549, 0.33595451, 0.33736642, 0.33877954,
        0.34019049, 0.34159532, 0.34299062, 0.34437443, 0.34574658,
        0.34710855, 0.34846289, 0.34981249, 0.35115974, 0.35250602,
        0.35385144, 0.35519496, 0.35653484, 0.35786916, 0.35919637]),
 array([[5.93119657, 6.25987121],
        [5.94212842, 6.36349956],
        [5.9976874 , 6.45117187],
        [6.06304087, 6.52178118],
        [6.11226291, 6.5710315 ],
        [6.13341519, 6.59377218],
        [6.12731109, 6.58838277],
        [6.10053458, 6.56225162],
        [6.06030619, 6.53164469],
        [6.0148553 , 6.514099  ],
        [5.97551741, 6.51924908],
```

```
       [5.95392877, 6.54657461],
       [5.95607097, 6.58944568],
       [5.979455  , 6.63935119],
       [6.01484545, 6.6881844 ],
       [6.05057794, 6.72919519],
       [6.07704893, 6.75749874],
       [6.08936183, 6.7708593 ],
       [6.08744092, 6.77069288],
       [6.0745819 , 6.76226786],
       [6.05592023, 6.75320281],
       [6.03740649, 6.75066572],
       [6.02480362, 6.75885791],
       [6.02224738, 6.77814257],
       [6.03086676, 6.80580699],
       [6.04840073, 6.83750737],
       [6.07014816, 6.86860052],
       [6.0907585 , 6.89512121],
       [6.10599332, 6.91448905],
       [6.11376618, 6.92601446],
       [6.1142786 , 6.93107291],
       [6.10952495, 6.93272268],
       [6.10254982, 6.93474057],
       [6.09666475, 6.94040888],
       [6.09466745, 6.95157083],
       [6.09814565, 6.96830142],
       [6.10707662, 6.98918065],
       [6.11991989, 7.01191116],
       [6.13419706, 7.03400904],
       [6.1473236 , 7.05340712],
       [6.15736597, 7.06889927],
       [6.16348167, 7.08037384],
       [6.16597657, 7.08877906],
       [6.16606078, 7.09580161],
       [6.16543521, 7.10333837],
       [6.16582225, 7.11293724],
       [6.16853041, 7.12539948],
       [6.17414254, 7.14065409],
       [6.18241105, 7.15789529],
       [6.19239468, 7.1758902 ],
       [6.20278847, 7.19334392],
       [6.21232813, 7.20922974],
       [6.22013388, 7.2230206 ],
       [6.22589929, 7.2347802 ],
       [6.22989883, 7.24509717],
       [6.23284515, 7.25488219],
       [6.23565617, 7.26509004],
       [6.23919792, 7.27645369],
       [6.24406424, 7.28930917],
       [6.25044431, 7.30355193],
       [6.25810854, 7.31871763],
       [6.26651069, 7.33414618],
       [6.2749691 , 7.34917458],
       [6.28286699, 7.36330799],
       [6.28981171, 7.37633078],
       [6.29571366, 7.38833594],
       [6.30077464, 7.39967134],
       [6.30540181, 7.41082233],
       [6.31007884, 7.42226768],
       [6.31523119, 7.43435154],
       [6.32111984, 7.44720593],
       [6.32778808, 7.46073932],
```

```
       [6.33507136, 7.47468615],
       [6.34266249, 7.48869601],
       [6.35020915, 7.50243451],
       [6.35741346, 7.51566905],
       [6.36410573, 7.52831932],
       [6.3702753 , 7.54046315],
       [6.37605576, 7.55230009],
       [6.38167387, 7.56408607],
       [6.38737972, 7.57605935],
       [6.39337824, 7.58837892],
       [6.39977948, 7.60109076],
       [6.40657897, 7.61412798],
       [6.41367063, 7.62734086],
       [6.42088557, 7.64054534],
       [6.42804387, 7.65357513],
       [6.43500375, 7.66632367],
       [6.44169526, 7.67876602],
       [6.44813101, 7.6909571 ],
       [6.45439389, 7.70300898],
       [6.4606075 , 7.71505522],
       [6.46689909, 7.7272132 ],
       [6.4733654 , 7.73955471],
       [6.48005008, 7.75209176],
       [6.48693762, 7.76477989],
       [6.49396352, 7.77753602],
       [6.50103665, 7.79026453],
       [6.50806626, 7.8028839 ],
       [6.5149861 , 7.81534677],
       [6.5217694 , 7.82764893],
       [6.52843172, 7.83982597],
       [6.53502226, 7.85193973],
       [6.54160716, 7.86405923],
       [6.5482502 , 7.8762416 ],
       [6.55499601, 7.88851823],
       [6.56186021, 7.90088925],
       [6.56882829, 7.91332682],
       [6.57586255, 7.92578552],
       [6.58291456, 7.93821626],
       [6.58993917, 7.95057967],
       [6.59690611, 7.96285555],
       [6.60380644, 7.97504621],
       [6.61065262, 7.98717351],
       [6.61747291, 7.99927112],
       [6.62430216, 8.01137429],
       [6.63117181, 8.02351047],
       [6.6381019 , 8.03569279],
       [6.64509691, 8.04791822],
       [6.65214614, 8.06017002]]))
```

In [ ]:

```
predictions = []
for x in np.arange(-5, 0):
    model = ARIMA(df_log.iloc[:x], order=(2, 1, 2))
    results_AR = model.fit()
    predictions.append(results_AR.forecast()[0])
predictions=np.array(predictions).ravel()
predictions
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:1
65: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  % freq, ValueWarning)
```

Out[ ]:

```
array([6.39078517, 6.34745416, 6.20273937, 6.181387  , 6.00689881])
```

In [ ]:

```
actual = df_log.tail()
actual
```

Out[ ]:

Month	#Passengers
1960-08-01	6.406880
1960-09-01	6.230481
1960-10-01	6.133398
1960-11-01	5.966147
1960-12-01	6.068426

In [ ]:

```
from sklearn.metrics import mean_squared_error

error = mean_squared_error(actual, predictions)
print('Test MSE: %.6f' % error)
```

```
Test MSE: 0.013773
```

In [ ]:

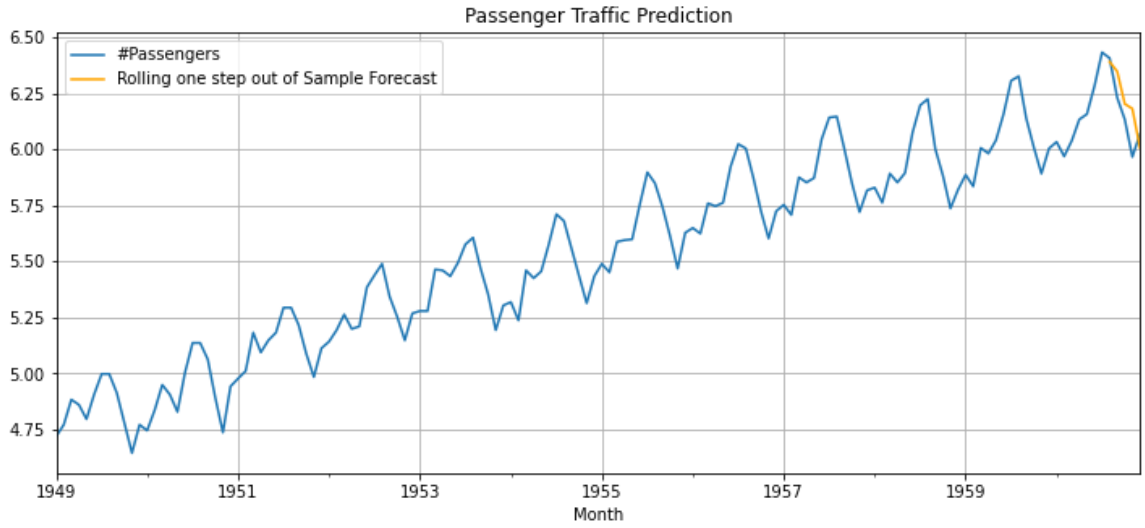
```
predictions_series = pd.Series(predictions, index = actual.index)
predictions_series
```

Out[ ]:

```
Month
1960-08-01    6.390785
1960-09-01    6.347454
1960-10-01    6.202739
1960-11-01    6.181387
1960-12-01    6.006899
dtype: float64
```

In [ ]:

```
#Future Forecasting Using the ARIMA Model

df_log.plot(figsize=(12, 5), grid=True, label= 'observed')
predictions_series.plot(c='orange', figsize=(12, 5), grid=True,label = 'Rolling one ste
p out of Sample Forecast')
plt.title('Passenger Traffic Prediction')
plt.legend()
plt.show()
```



Questions 11-15:

Scenario 3 - Minimum Daily Temperature Analysis

The **Bureau of Meteorology** (Australian Weather Department) is an Executive Agency of the **Australian Government** responsible for providing weather services to Australia and surrounding areas with some basic data of Daily Minimum Temperatures over 10 years, i.e., from 1981 - 1990 in the city of Melbourne, Australia.

In Time Series Forecasting, we use a model to predict future values based on previously observed values. It is a useful method because the Australian Weather Department can use it to predict/forecast future weather conditions in the city of Melbourne.

Problem Statement

The Bureau of Metrology has the data of minimum daily temperatures of the city of Melbourne, Australia. The data is classified into two columns i.e., date and the minimum temperature.

So, being a Data Scientist, you must -

Build a model to forecast the minimum daily temperatures in the future.

Dataset Description

This dataset describes the minimum daily temperatures over 10 years (1981-1990) in the city of Melbourne, Australia.

The units are in degrees Celsius, and there are 3650 observations. The source of the data is credited as the Australian Bureau of Meteorology.

Here's a brief description of the dataset:

- **Date** - A Date
- **Temp** - Minimum Temperature

Tasks to be performed

- Load, analyze, pre-process and visualize the dataset to determine the trend - Beginner
- Check the stationarity of the dataset using the Rolling Statistics technique and plot the Rolling Statistics - Intermediate
- Check the stationarity of the dataset using the Dickey-Fuller Test - Intermediate
- Plot the Auto-Correlation and Partial Auto-Correlation Graph - Intermediate
- Build the ARIMA Model and forecast the demand - Advanced

Topics Covered

- Rolling Statistics
- Dickey-Fuller Test
- Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF)
- ARIMA Model

In [ ]:

```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/3tvhg63inpw052u/daily-min-temperatures.csv
```

```
--2020-06-29 13:00:31--  https://www.dropbox.com/s/3tvhg63inpw052u/daily-m
in-temperatures.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:
1::a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connec
ted.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/3tvhg63inpw052u/daily-min-temperatures.csv [following]
--2020-06-29 13:00:31--  https://www.dropbox.com/s/raw/3tvhg63inpw052u/dai
ly-min-temperatures.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc78622362b3a169fe553c879d6a.dl.dropboxusercontent.com/c
d/0/inline/A6kd2Hk7L3G12YWzYYlLjXiTWRcqxCfjErcVWAzdTj6JWm97biNSPCMadrK38bM
GmZPgp34uWxgdg6jDDZWUjBdLGlFNO-H_NM37pmD8apXrwnRNgsymu3GzfAbEMYldDUc/file#
[following]
--2020-06-29 13:00:32--  https://uc78622362b3a169fe553c879d6a.dl.dropboxus
ercontent.com/cd/0/inline/A6kd2Hk7L3G12YWzYYlLjXiTWRcqxCfjErcVWAzdTj6JWm97
biNSPCMadrK38bMGmZPgp34uWxgdg6jDDZWUjBdLGlFNO-H_NM37pmD8apXrwnRNgsymu3GzfA
bEMYldDUc/file
Resolving uc78622362b3a169fe553c879d6a.dl.dropboxusercontent.com (uc786223
62b3a169fe553c879d6a.dl.dropboxusercontent.com)... 162.125.5.15, 2620:100:
601d:15::a27d:50f
Connecting to uc78622362b3a169fe553c879d6a.dl.dropboxusercontent.com (uc78
622362b3a169fe553c879d6a.dl.dropboxusercontent.com)|162.125.5.15|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 67921 (66K) [text/plain]
Saving to: ‘daily-min-temperatures.csv’

daily-min-temperatu 100%[===================>]  66.33K  --.-KB/s    in 0.0
5s

2020-06-29 13:00:32 (1.23 MB/s) - ‘daily-min-temperatures.csv’ saved [6792
1/67921]
```

Question 11 - Beginner

Load, analyze, pre-process and visualize the dataset to determine the trend.

In [ ]:

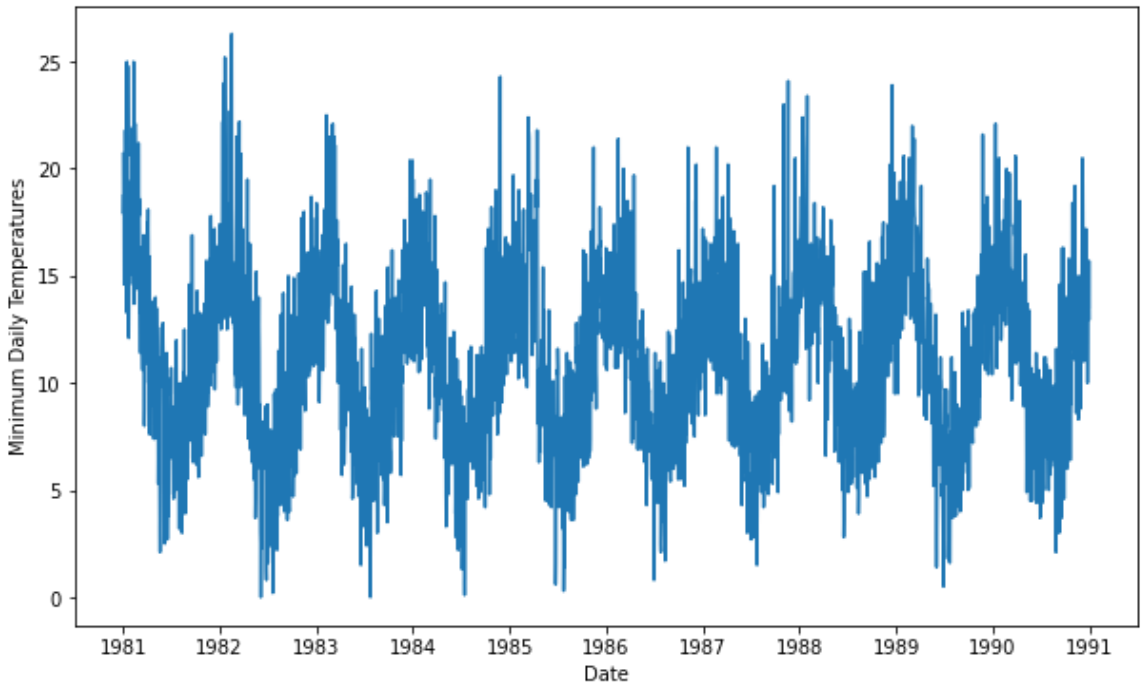
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 10,6

df=pd.read_csv('/content/daily-min-temperatures.csv', parse_dates=True, index_col='Dat
e')
df.head()
```

Out[ ]:

Date	Temp
1981-01-01	20.7
1981-01-02	17.9
1981-01-03	18.8
1981-01-04	14.6
1981-01-05	15.8

In [ ]:

```
plt.xlabel('Date')
plt.ylabel('Minimum Daily Temperatures')
plt.plot(df)
plt.show()
```



From the above, you can see that the data has no particular trend. It is **Stationary**.

Question 12 - Intermediate

Check the stationarity of the dataset using the Rolling Statistics technique and plot the Rolling Statistics.

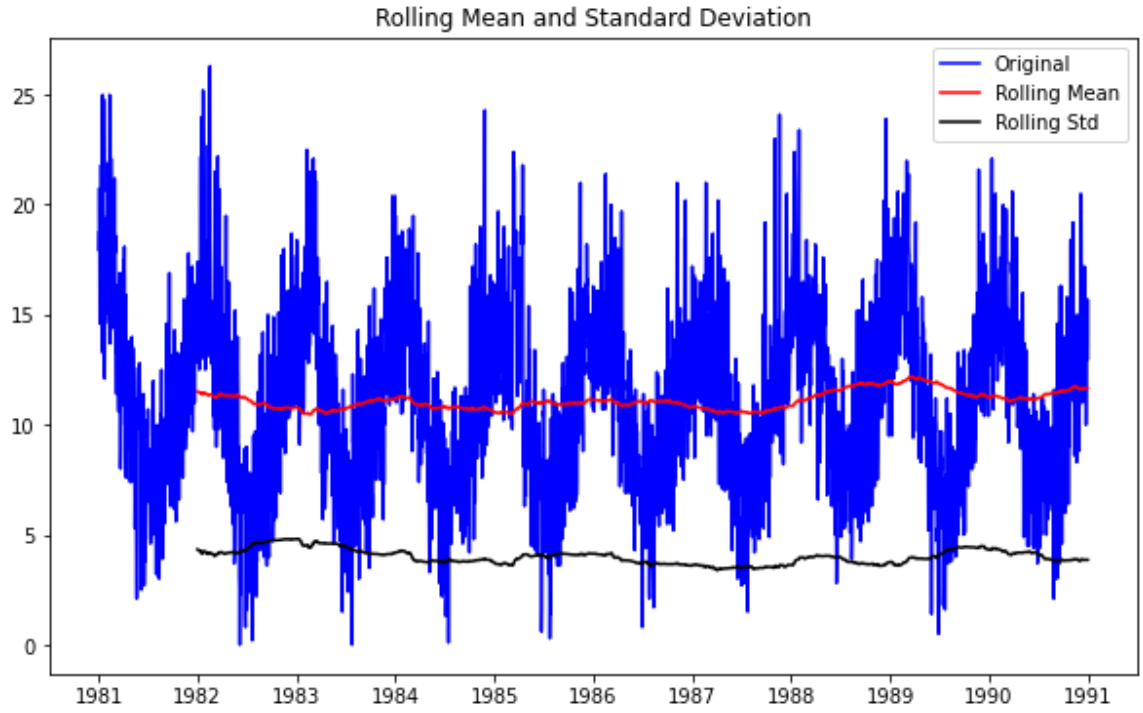
In [ ]:

```
#Determine Rolling Statistics
rollmean = df.rolling(window = 365).mean() #Rolling Mean at daily level
rollstd = df.rolling(window = 365).std()

print(rollmean, rollstd)
```

```
                Temp
Date
1981-01-01       NaN
1981-01-02       NaN
1981-01-03       NaN
1981-01-04       NaN
1981-01-05       NaN
...              ...
1990-12-27  11.651507
1990-12-28  11.656712
1990-12-29  11.665205
1990-12-30  11.668767
1990-12-31  11.669589

[3650 rows x 1 columns]                Temp
Date
1981-01-01      NaN
1981-01-02      NaN
1981-01-03      NaN
1981-01-04      NaN
1981-01-05      NaN
...             ...
1990-12-27  3.856232
1990-12-28  3.857580
1990-12-29  3.858218
1990-12-30  3.861348
1990-12-31  3.861600

[3650 rows x 1 columns]
```

In [ ]:

```
#Plot Rolling Statistics

orig = plt.plot(df, color = 'blue', label = 'Original')
mean = plt.plot(rollmean, color = 'red', label = 'Rolling Mean')
std = plt.plot(rollstd, color = 'black', label = 'Rolling Std')
plt.legend(loc = 'best')
plt.title("Rolling Mean and Standard Deviation")
plt.show()
```



From the above, you can see that the **Rolling Mean** and **Standard Deviation** are constant. That means the dataset is **Stationary**.

Question 13 - Intermediate

Check the stationarity of the dataset using the Dickey-Fuller Test.

Null Hypothesis - Minimum Daily Temperatures Dataset is Non-Stationary

Alternate Hypothesis - Minimum Daily Temperatures Dataset is Stationary

In [ ]:

```
from statsmodels.tsa.stattools import adfuller

print("Results of Dickey Fuller Test:")

dftest = adfuller(df['Temp'], autolag = 'AIC')

dfoutput = pd.Series(dftest[0:4], index = ['Test Statistic', 'p-value', '#Lags Used',
'No. of Observations Used'])

for key, value in dftest[4].items():
  dfoutput['Critical Value: %s'%key] = value

print(dfoutput)
```

```
Results of Dickey Fuller Test:
Test Statistic                -4.444805
p-value                        0.000247
#Lags Used                    20.000000
No. of Observations Used    3629.000000
Critical Value: 1%            -3.432153
Critical Value: 5%            -2.862337
Critical Value: 10%           -2.567194
dtype: float64
```

From the above, you can see that the p-value is very less i.e. We **reject the Null Hypothesis** and conclude that the data is **Stationary**.

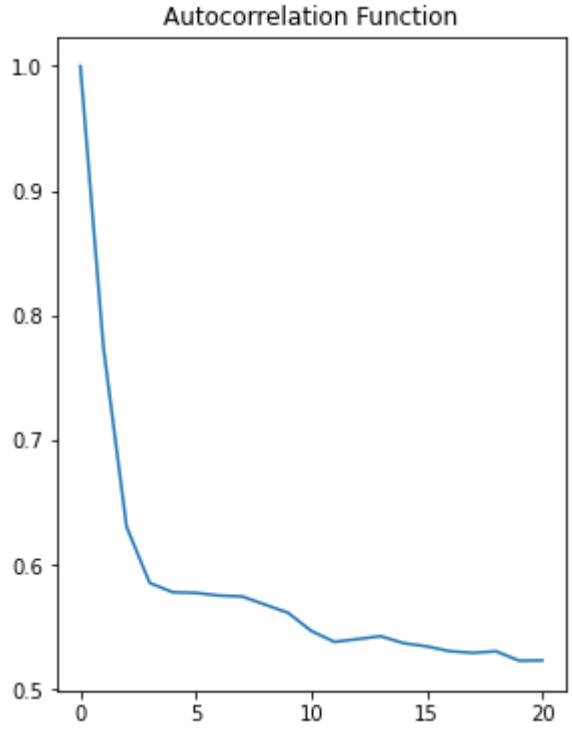
Question 14 - Intermediate

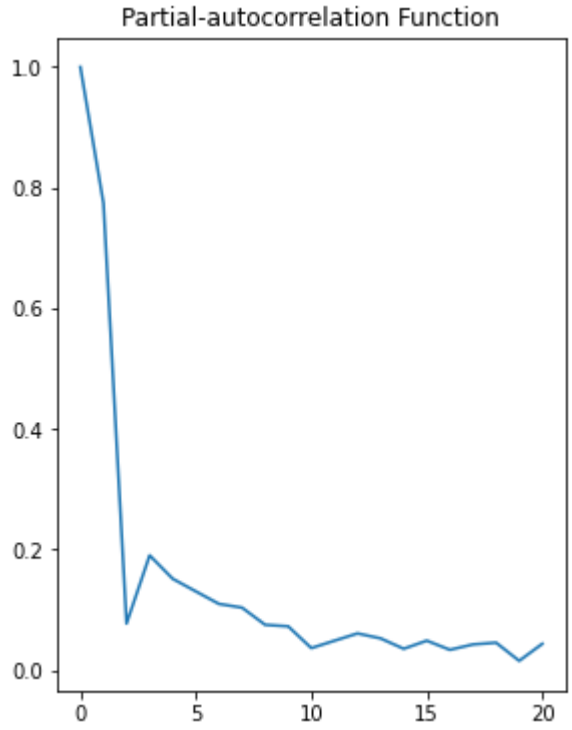
Plot the Auto-Correlation and Partial Auto-Correlation Graph.

In [ ]:

```
from statsmodels.tsa.stattools import acf, pacf

lag_acf = acf(df, nlags = 20)
lag_pacf = pacf(df, nlags = 20, method = 'ols')

#Plot ACF
plt.subplot(121)
plt.plot(lag_acf)
plt.title("Autocorrelation Function")
plt.show()

#Plot PACF
plt.subplot(122)
plt.plot(lag_pacf)
plt.title("Partial-autocorrelation Function")
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/stattools.py:541: F
utureWarning: fft=True will become the default in a future version of stat
smodels. To suppress this warning, explicitly set fft=False.
  warnings.warn(msg, FutureWarning)
```





Question 15 - Advanced

Build the ARIMA Model and forecast the demand.

In [ ]:

```
from statsmodels.tsa.arima_model import ARIMA

#AR Model
model = ARIMA(df, order = (2, 1, 2))
results_AR = model.fit()

results_AR.summary()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:2
19: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:2
19: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

Out[ ]:

ARIMA Model Results

Dep. Variable:	D.Temp	**No. Observations:**	3649
Model:	ARIMA(2, 1, 2)	**Log Likelihood**	-8386.371
Method:	css-mle	**S.D. of innovations**	2.409
Date:	Mon, 29 Jun 2020	**AIC**	16784.742
Time:	13:03:58	**BIC**	16821.955
Sample:	1	**HQIC**	16797.995

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0012	0.006	-0.193	0.847	-0.013	0.011
ar.L1.D.Temp	0.4853	0.139	3.490	0.000	0.213	0.758
ar.L2.D.Temp	-0.1243	0.067	-1.851	0.064	-0.256	0.007
ma.L1.D.Temp	-0.8896	0.140	-6.351	0.000	-1.164	-0.615
ma.L2.D.Temp	-0.0104	0.129	-0.080	0.936	-0.264	0.243

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.9521	-2.0577j	2.8364	-0.1292
AR.2	1.9521	+2.0577j	2.8364	0.1292
MA.1	1.1097	+0.0000j	1.1097	0.0000
MA.2	-86.7975	+0.0000j	86.7975	0.5000

In [ ]:

```
#Analyse residual errors. They should be normal
residuals = pd.DataFrame(results_AR.resid)
print(residuals.describe())
```

```
                0
count  3649.000000
mean     -0.001378
std       2.409638
min      -7.402870
25%      -1.595123
50%       0.045428
75%       1.536410
max      10.442989
```

In [ ]:

```
results_AR.forecast(steps = 100)
```

Out[ ]:

```
(array([13.08939937, 13.48742015, 13.66869526, 13.70642246, 13.70142841,
        13.69354522, 13.68957028, 13.68785112, 13.68674088, 13.68564574,
        13.68448225, 13.6832837 , 13.68207664, 13.68086981, 13.67966414,
        13.67845901, 13.677254  , 13.67604898, 13.67484394, 13.67363889,
        13.67243383, 13.67122878, 13.67002373, 13.66881868, 13.66761363,
        13.66640858, 13.66520353, 13.66399848, 13.66279343, 13.66158837,
        13.66038332, 13.65917827, 13.65797322, 13.65676817, 13.65556312,
        13.65435807, 13.65315302, 13.65194797, 13.65074292, 13.64953787,
        13.64833281, 13.64712776, 13.64592271, 13.64471766, 13.64351261,
        13.64230756, 13.64110251, 13.63989746, 13.63869241, 13.63748736,
        13.6362823 , 13.63507725, 13.6338722 , 13.63266715, 13.6314621 ,
        13.63025705, 13.629052  , 13.62784695, 13.6266419 , 13.62543685,
        13.62423179, 13.62302674, 13.62182169, 13.62061664, 13.61941159,
        13.61820654, 13.61700149, 13.61579644, 13.61459139, 13.61338634,
        13.61218129, 13.61097623, 13.60977118, 13.60856613, 13.60736108,
        13.60615603, 13.60495098, 13.60374593, 13.60254088, 13.60133583,
        13.60013078, 13.59892572, 13.59772067, 13.59651562, 13.59531057,
        13.59410552, 13.59290047, 13.59169542, 13.59049037, 13.58928532,
        13.58808027, 13.58687521, 13.58567016, 13.58446511, 13.58326006,
        13.58205501, 13.58084996, 13.57964491, 13.57843986, 13.57723481]),
 array([2.40892141, 2.80392741, 2.87556418, 2.89953812, 2.9196611 ,
        2.94186071, 2.96546745, 2.98937294, 3.0131258 , 3.03665099,
        3.05997066, 3.08310718, 3.10607114, 3.12886727, 3.15149889,
        3.17396923, 3.1962816 , 3.21843928, 3.24044545, 3.26230317,
        3.28401542, 3.30558506, 3.32701486, 3.34830751, 3.3694656 ,
        3.39049167, 3.41138814, 3.43215739, 3.45280171, 3.47332332,
        3.4937244 , 3.51400704, 3.53417328, 3.5542251 , 3.57416443,
        3.59399314, 3.61371304, 3.63332592, 3.6528335 , 3.67223744,
        3.6915394 , 3.71074095, 3.72984366, 3.74884902, 3.76775853,
        3.7865736 , 3.80529564, 3.82392602, 3.84246607, 3.86091709,
        3.87928035, 3.8975571 , 3.91574854, 3.93385586, 3.95188022,
        3.96982273, 3.98768452, 4.00546665, 4.02317019, 4.04079617,
        4.0583456 , 4.07581946, 4.09321873, 4.11054435, 4.12779725,
        4.14497834, 4.16208851, 4.17912862, 4.19609954, 4.21300209,
        4.2298371 , 4.24660538, 4.2633077 , 4.27994484, 4.29651755,
        4.31302659, 4.32947268, 4.34585653, 4.36217884, 4.37844031,
        4.39464161, 4.41078339, 4.42686632, 4.44289103, 4.45885815,
        4.4747683 , 4.49062207, 4.50642008, 4.52216289, 4.53785109,
        4.55348523, 4.56906589, 4.58459359, 4.60006888, 4.61549228,
        4.63086431, 4.64618548, 4.6614563 , 4.67667725, 4.69184883]),
 array([[ 8.36800017, 17.81079858],
        [ 7.9918234 , 18.9830169 ],
        [ 8.03269303, 19.30469749],
        [ 8.02343217, 19.38941274],
        [ 7.97899781, 19.42385901],
        [ 7.92760418, 19.45948626],
        [ 7.87736088, 19.50177968],
        [ 7.82878782, 19.54691443],
        [ 7.78112284, 19.59235892],
        [ 7.73391916, 19.63737232],
        [ 7.68704997, 19.68191453],
        [ 7.64050466, 19.72606274],
        [ 7.59428908, 19.7698642 ],
        [ 7.54840265, 19.81333697],
        [ 7.50283983, 19.85648845],
        [ 7.45759364, 19.89932438],
        [ 7.41265718, 19.94185082],
        [ 7.36802391, 19.98407405],
        [ 7.32368757, 20.0260003 ],
```

```
       [ 7.27964216, 20.06763561],
       [ 7.23588189, 20.10898578],
       [ 7.19240112, 20.15005644],
       [ 7.14919444, 20.19085303],
       [ 7.10625656, 20.2313808 ],
       [ 7.0635824 , 20.27164486],
       [ 7.02116702, 20.31165013],
       [ 6.97900564, 20.35140142],
       [ 6.93709361, 20.39090334],
       [ 6.89542644, 20.43016041],
       [ 6.85399975, 20.46917699],
       [ 6.81280932, 20.50795732],
       [ 6.77185103, 20.54650552],
       [ 6.73112087, 20.58482557],
       [ 6.69061498, 20.62292136],
       [ 6.65032956, 20.66079668],
       [ 6.61026096, 20.69845518],
       [ 6.5704056 , 20.73590043],
       [ 6.53076002, 20.77313592],
       [ 6.49132082, 20.81016501],
       [ 6.45208473, 20.846991  ],
       [ 6.41304855, 20.88361708],
       [ 6.37420914, 20.92004639],
       [ 6.33556347, 20.95628195],
       [ 6.29710859, 20.99232673],
       [ 6.2588416 , 21.02818362],
       [ 6.22075969, 21.06385543],
       [ 6.18286011, 21.09934491],
       [ 6.14514018, 21.13465473],
       [ 6.1075973 , 21.16978751],
       [ 6.07022891, 21.2047458 ],
       [ 6.03303253, 21.23953208],
       [ 5.99600571, 21.2741488 ],
       [ 5.95914609, 21.30859832],
       [ 5.92245134, 21.34288296],
       [ 5.88591921, 21.37700499],
       [ 5.84954747, 21.41096663],
       [ 5.81333396, 21.44477004],
       [ 5.77727657, 21.47841733],
       [ 5.74137321, 21.51191058],
       [ 5.70562188, 21.54525181],
       [ 5.67002059, 21.578443  ],
       [ 5.63456739, 21.6114861 ],
       [ 5.5992604 , 21.64438299],
       [ 5.56409775, 21.67713553],
       [ 5.52907764, 21.70974554],
       [ 5.49419827, 21.74221481],
       [ 5.45945791, 21.77454507],
       [ 5.42485485, 21.80673803],
       [ 5.39038742, 21.83879536],
       [ 5.35605397, 21.8707187 ],
       [ 5.3218529 , 21.90250967],
       [ 5.28778264, 21.93416983],
       [ 5.25384164, 21.96570072],
       [ 5.2200284 , 21.99710387],
       [ 5.18634142, 22.02838075],
       [ 5.15277924, 22.05953282],
       [ 5.11934046, 22.0905615 ],
       [ 5.08602365, 22.12146821],
       [ 5.05282745, 22.15225431],
       [ 5.01975051, 22.18292115],
```

```
       [ 4.9867915 , 22.21347005],
       [ 4.95394913, 22.24390232],
       [ 4.92122212, 22.27421923],
       [ 4.88860921, 22.30442204],
       [ 4.85610918, 22.33451196],
       [ 4.82372082, 22.36449022],
       [ 4.79144294, 22.394358  ],
       [ 4.75927437, 22.42411647],
       [ 4.72721397, 22.45376676],
       [ 4.69526062, 22.48331002],
       [ 4.6634132 , 22.51274733],
       [ 4.63167063, 22.5420798 ],
       [ 4.60003185, 22.57130848],
       [ 4.56849579, 22.60043444],
       [ 4.53706143, 22.6294587 ],
       [ 4.50572775, 22.65838227],
       [ 4.47449375, 22.68720617],
       [ 4.44335845, 22.71593137],
       [ 4.41232087, 22.74455884],
       [ 4.38138008, 22.77308953]]))
```

Module 5 - Model Evaluation and Hyperparameter Tuning

and

Module 6 - Model Boosting and Optimization

- **Model selection** is the process of selecting one final machine learning model from among a collection of can
- The chosen model is then tested on new (test) data to check its Generalization ability
- Generalization refers to the ability of a selected model to provide consistent performance on unseen data - thi
- In machine learning, after we have chosen a model, we can optimize it for our problem by **tuning the model**
- Model hyperparameters can be thought of as settings for a machine learning algorithm that are set by the dat

In [ ]:

```
## Getting the required Datasets:
!wget https://www.dropbox.com/s/5x30kmmqk9gdfrf/concrete.csv -nv
!wget https://www.dropbox.com/s/ahfj6zfskjirxei/Depressed.csv -nv
!wget https://www.dropbox.com/s/cuh3asfjpj9fgzi/housevotes.csv -nv
!wget https://www.dropbox.com/s/v10zn2rodytj5mb/NBA.csv -nv
```

```
2020-07-16 08:18:20 URL:https://ucc0f85eb0066ab905541f614391.dl.dropboxuse
rcontent.com/cd/0/inline/A7mXInu8CgBR7eHSj4nYlciFVqK60SiA7FyZ3lpy9hgyWNDkS
LSpHto1FTDveIyjxIOYKBZ9ZTFvTa2S7k9eSmDxnyAbe8p2cTy9wSpV6RJtSZ070UCCW_GGi0e
9S752MO4/file [105237/105237] -> "concrete.csv" [1]
2020-07-16 08:18:23 URL:https://ucad68a51c37beb5199ab6ed85bd.dl.dropboxuse
rcontent.com/cd/0/inline/A7mdAFtO_CQrsyLt1ll28-4lHwimZ7PK0GDHII0DC_RFRMbjZ
un4NYn3khx97_MR2isxvG3JXXx-r9kji2Tx3AmOQFAadPerxitP2Nqo9TBfPtSqb4uR2l3HtFT
as0ySg1s/file [191857/191857] -> "Depressed.csv" [1]
2020-07-16 08:18:25 URL:https://uce423f0c8fd59774d279d74bef1.dl.dropboxuse
rcontent.com/cd/0/inline/A7mgsJ8EhkvjyT8r8jnN5ok62AzXkokvZuERnx4r9Nl_9Plaj
jCNDoIVztgw7LvqVfMIp7zlQ38lERQmpvY3iU_w1GdrBq5LQa605K1k5KQ8DVaXNBHL_1u98X0
DWUbVd_w/file [17779/17779] -> "housevotes.csv" [1]
2020-07-16 08:18:27 URL:https://ucb1b8a32458c25f511f1e39fac0.dl.dropboxuse
rcontent.com/cd/0/inline/A7kQs9M6JKoLyrSmROFjqZ3BE9Quao6aE4m3rGS_XiE_skis2
hug_alr1gmpNyQ8heKYlaYqawbEjP5KyyaHiAFRpjYslx5aGEwjYMUjg4hjFt5FP95GKTSrdPL
DvpTF-2Y/file [129360/129360] -> "NBA.csv" [1]
```

Questions 1-5:

Scenario 1 - Congressional Votes

Based on 16 key votes identified by CQA, the data collected has votes from each U.S. House of Representatives Congressmen. There are 9 different types of votes listed by CQA: voted against, paired against, voted for, paired for, announced for, announced against, voted present, voted present to avoid conflict of interest, and no vote or otherwise.

Problem Statement:

Based on the votes, the aim is to classify either a congressman is a republican or a democrat.

Dataset Description:

housevotes.csv - Here's a brief description of the attributes in the dataset:

Attributes:

- **Handicapped_infants**:{y,n}
- **Water_project_cost**:{y,n}
- **Adoption_budget_resolution**:{y,n}
- **Physician_fee_freeze**:{y,n}
- **El_salvador_aid**:{y,n}
- **Religious_groups_in_schools**:{y,n}
- **Anti_satellite_test_ban**:{y,n}
- **Aid_to_nicaraguan_contras**:{y,n}
- **Mx_missile**:{y,n}
- **Immigration**:{y,n}
- **Synfuels_corporation_cutback**:{y,n}
- **Education_spending**:{y,n}
- **Superfund_right_to_sue**:{y,n}
- **Crime**:{y,n}
- **Duty_free_exports**:{y,n}
- **Export_south_africa**:{y,n}

Target Variable:

- **Class**: {republican,democrat}

Tasks to be performed:

In order to attain the above goal, the below tasks must be performed:

- Read the dataset with no headers, then put respective columns names - **Beginner**
- Preprocess the data - **Beginner**
- Split the data into training and testing set and apply Bernoulli Naive Bayes and Logistic Regression models - **Intermediate**
- Evaluate the models using confusion matrix - **Intermediate**
- Using LOOCV, fit a Logistic Regression model and Bernoulli model; Calculate the average score using accuracy as scoring parameter - **Advanced**

Topics covered:

- Naive Bayes
- Logistic Regression
- LOOCV

Question 1 - Beginner

Read the dataset with no headers, then put respective columns names.

In [ ]:

```
import pandas as pd

cols=['Handicapped_infants','Water_project_cost','Adoption_budget_resolution','Physician_fee_freeze',
'El_salvador_aid','Religious_groups_in_schools','Anti_satellite_test_ban','Aid_to_nicaraguan_contras',
'Mx_missile','Immigration','Synfuels_corporation_cutback','Education_spending','Superfund_right_to_sue',
'Crime','Duty_free_exports','Export_south_africa','Target']

data=pd.read_csv('housevotes.csv',header=None)
data.columns=cols
data.head()
```

Out[ ]:

	Handicapped_infants	Water_project_cost	Adoption_budget_resolution	Physician_fee_freeze
0	n	y	n	y
1	n	y	n	y
2	NaN	y	y	NaN
3	n	y	y	n
4	y	y	y	n

In [ ]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435 entries, 0 to 434
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Handicapped_infants             423 non-null    object
 1   Water_project_cost              387 non-null    object
 2   Adoption_budget_resolution      424 non-null    object
 3   Physician_fee_freeze            424 non-null    object
 4   El_salvador_aid                 420 non-null    object
 5   Religious_groups_in_schools     424 non-null    object
 6   Anti_satellite_test_ban         421 non-null    object
 7   Aid_to_nicaraguan_contras       420 non-null    object
 8   Mx_missile                      413 non-null    object
 9   Immigration                     428 non-null    object
 10  Synfuels_corporation_cutback    414 non-null    object
 11  Education_spending              404 non-null    object
 12  Superfund_right_to_sue          410 non-null    object
 13  Crime                           418 non-null    object
 14  Duty_free_exports               407 non-null    object
 15  Export_south_africa             331 non-null    object
 16  Target                          435 non-null    object
dtypes: object(17)
memory usage: 57.9+ KB
```

In [ ]:

```
data.describe(include='all')
```

Out[ ]:

	Handicapped_infants	Water_project_cost	Adoption_budget_resolution	Physician_fee_f
count	423	387	424	
unique	2	2	2	
top	n	y	y	
freq	236	195	253	

Question 2 - Beginner

Preprocess the data.

In [ ]:

```
# Acquire the datatset and sort it
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in dat
a.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	Handicapped_infants	False	0	0.0
9	Immigration	False	0	0.0
15	Export_south_africa	False	0	0.0
14	Duty_free_exports	False	0	0.0
13	Crime	False	0	0.0
12	Superfund_right_to_sue	False	0	0.0
11	Education_spending	False	0	0.0
10	Synfuels_corporation_cutback	False	0	0.0
8	Mx_missile	False	0	0.0
1	Water_project_cost	False	0	0.0
7	Aid_to_nicaraguan_contras	False	0	0.0
6	Anti_satellite_test_ban	False	0	0.0
5	Religious_groups_in_schools	False	0	0.0
4	El_salvador_aid	False	0	0.0
3	Physician_fee_freeze	False	0	0.0
2	Adoption_budget_resolution	False	0	0.0
16	Target	False	0	0.0

In [ ]:

```
# Count the total missing values
print('Total Missing values: %s'%sum(miss.Count_))
```

```
Total Missing values: 0
```

In [ ]:

```
# Handle the missing values
import numpy as np

for i in miss[miss.Count_!=0].Col_name:
    temp=data[i].mode()
    tes=data[i].isnull()
    ind=tes[tes==True].index
    quill=list(data[i])
    for j in ind:
        quill[j]=temp[0]
    data[i]=quill

miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                        [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                        'Percentage':[sum(data[y].isnull())/data.shape[0] for y in dat
a.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	Handicapped_infants	False	0	0.0
9	Immigration	False	0	0.0
15	Export_south_africa	False	0	0.0
14	Duty_free_exports	False	0	0.0
13	Crime	False	0	0.0
12	Superfund_right_to_sue	False	0	0.0
11	Education_spending	False	0	0.0
10	Synfuels_corporation_cutback	False	0	0.0
8	Mx_missile	False	0	0.0
1	Water_project_cost	False	0	0.0
7	Aid_to_nicaraguan_contras	False	0	0.0
6	Anti_satellite_test_ban	False	0	0.0
5	Religious_groups_in_schools	False	0	0.0
4	El_salvador_aid	False	0	0.0
3	Physician_fee_freeze	False	0	0.0
2	Adoption_budget_resolution	False	0	0.0
16	Target	False	0	0.0

In [ ]:

```
data.head()
```

Out[ ]:

	Handicapped_infants	Water_project_cost	Adoption_budget_resolution	Physician_fee_freeze
0	n	y	n	y
1	n	y	n	y
2	n	y	y	n
3	n	y	y	n
4	y	y	y	n

In [ ]:

```
# Encoding target labels using LabelEncoder
from sklearn.preprocessing import LabelEncoder

def lb_encode(x):
    lb=LabelEncoder()
    return lb.fit_transform(x)

label_data=data.apply(lb_encode)
label_data.head()
```

Out[ ]:

	Handicapped_infants	Water_project_cost	Adoption_budget_resolution	Physician_fee_freeze
0	0	1	0	1
1	0	1	0	1
2	0	1	1	0
3	0	1	1	0
4	1	1	1	0

Question 3 - Intermediate (Bridging Question)

Split the data into training and testing set and apply Bernoulli Naive Bayes and Logistic Regression models.

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression

X=label_data.drop('Target',axis=1)
y=label_data.Target
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75,test_size=0.2
5, random_state=101)

bnb=BernoulliNB(alpha=5)
bnb.fit(X_train,y_train)
```

Out[ ]:

```
BernoulliNB(alpha=5, binarize=0.0, class_prior=None, fit_prior=True)
```

In [ ]:

```
lgm=LogisticRegression(C=2)
lgm.fit(X_train,y_train)
```

Out[ ]:

```
LogisticRegression(C=2, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [ ]:

```
b_pred=bnb.predict(X_test)
```

In [ ]:

```
l_pred=lgm.predict(X_test)
```

Question 4 - Intermediate

Evaluate the models using confusion matrix.

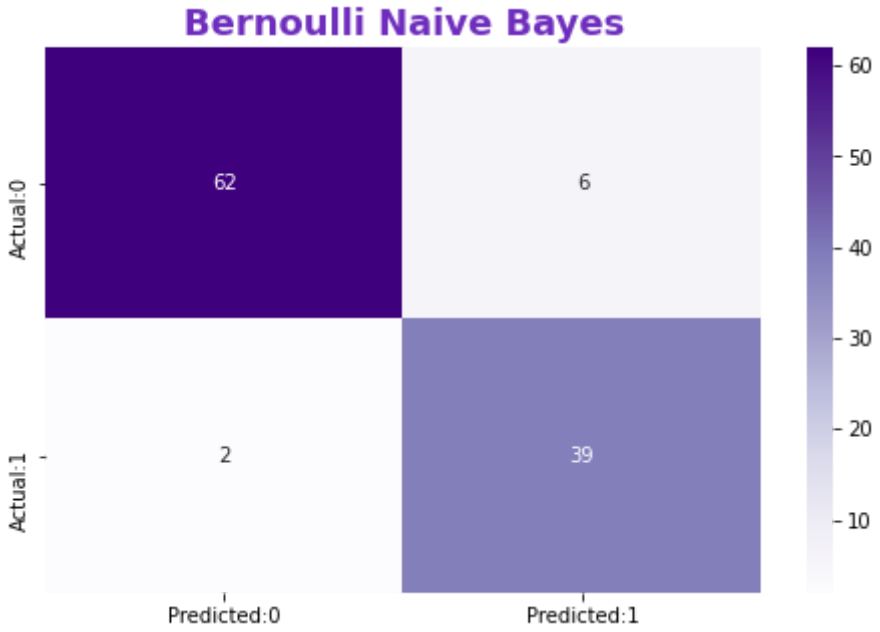
In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

print('The accuracy of the bernoulli  model is: ',round(accuracy_score(y_test,b_pred)*1
00,2))
print('The accuracy of the logistic  model is: ',round(accuracy_score(y_test,l_pred)*10
0,2))
```

```
The accuracy of the bernoulli  model is:  92.66
The accuracy of the logistic  model is:  96.33
```

In [ ]:

```
dcm=confusion_matrix(y_test,b_pred)
dconf_matrix=pd.DataFrame(data=dcm,columns=['Predicted:0','Predicted:1'],index=['Actua
l:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(dconf_matrix, annot=True,fmt='d',cmap='Purples')
fm={'size':18,'color':'#6f2dbd','weight':'bold'}
plt.title('Bernoulli Naive Bayes',**fm)
plt.show()
```



In [ ]:

```
TN=dcm[0,0]
TP=dcm[1,1]
FN=dcm[1,0]
FP=dcm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Posetive", TP)
print("False Negative", FN)
print("False Posetive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 62
True Posetive 39
False Negative 2
False Posetive 6
Sensitivity 0.9512195121951219
Specificity 0.9117647058823529
```

We see that Bernoulli model is slightly more sensitive.

In [ ]:

```
dcm=confusion_matrix(y_test,l_pred)
dconf_matrix=pd.DataFrame(data=dcm,columns=['Predicted:0','Predicted:1'],index=['Actua
l:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(dconf_matrix, annot=True,fmt='d',cmap='Greens')
fm={'size':18,'color':'#38a700','weight':'bold'}
plt.title('Logistic Regression',**fm)
plt.show()
```

In [ ]:

```
TN=dcm[0,0]
TP=dcm[1,1]
FN=dcm[1,0]
FP=dcm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Posetive", TP)
print("False Negative", FN)
print("False Posetive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 66
True Posetive 39
False Negative 2
False Posetive 2
Sensitivity 0.9512195121951219
Specificity 0.9705882352941176
```

We see that overall, Logistic model is performing better than Bernoulli model.

Question 5 - Advanced

Using LOOCV, fit a Logistic Regression model and Bernoulli model; Calculate the average score using accuracy as scoring parameter.

In [ ]:

```
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import accuracy_score, mean_absolute_error


#the leave one out
loo = LeaveOneOut()
loo.get_n_splits(X)
scores=[]
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index,:], X.iloc[test_index,:]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    model=LogisticRegression(C=2)
    model.fit(X_train,y_train)
    scores.append(mean_absolute_error(y_test,model.predict(X_test)))

print('Average Score: %.2f'%np.mean(scores))
```

```
Average Score: 0.04
```

In [ ]:

```
loo = LeaveOneOut()
loo.get_n_splits(X)
scores=[]
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index,:], X.iloc[test_index,:]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    model=BernoulliNB(alpha=5)
    model.fit(X_train,y_train)
    scores.append(mean_absolute_error(y_test,model.predict(X_test)))

print('Average Score: %.2f'%np.mean(scores))
```

```
Average Score: 0.10
```

Questions 6-11:

Scenario 2 - Crayola Cement Foundation

Established in 1996, Crayola Cement Foundation is one of the top competitors in the cement industries. They are proud of an exceptional record of efficiency, safety, and productivity while always seeking the newest techniques, systems, and equipment. They make precast and prestressed concrete products such as columns, raker beams, wall panels, spandrels, seat risers, double and inverted tees, hollow-core, and stairs.

Problem Statement:

Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate. CCF wants to find out the compressive strength of the concrete based on these attributes.

Dataset Description:

concrete.csv - Here's a brief description of the attributes in the dataset:

Attributes:

- **Cement**: real [102.0, 540.0]
- **BlastFurnaceSlag**: real [0.0, 359.4]
- **FlyAsh**: real [0.0, 200.100006]
- **Water**: real [121.8, 247.0]
- **Superplasticizer**: real [0.0, 32.200001]
- **CoarseAggregate**: real [801.0, 1145.0]
- **FineAggregate**: real [594.0, 992.6]
- **Age**: integer [1, 365]

Target Variable:

- **ConcreteCompressiveStrength**: real [2.33, 82.6]

Tasks to be performed:

To attain the above goal, the below tasks must be performed:

- Read the dataset with no headers, then put respective column names and find the correlation between the features - **Beginner**
- Fit a Linear Regression model by applying 5-fold cross validation using scikit and calculate the scores and average accuracy - **Intermediate**
- Fit a Linear Regression model by applying 5-fold cross validation explicitly using KFold scikit and calculate the scores using mean absolute error as scoring parameter - **Advanced**
- Using LOOCV, fit a Linear Regression model and calculate the average score using mean absolute error as scoring parameter **Advanced**
- Apply Lasso Regression over the model and find R2_score and MAE - **Beginner**
- Apply Ridge Regression over the model and find R2_score and MAE - **Beginner**

Topics covered:

- Linear Regression
- Lasso Regression
- Ridge Regression
- K-Fold
- RMSE
- Mean Absolute Error
- R-Square
- LOOCV

Question 6 - Beginner

Read the dataset with no headers, then put respective column names and find the correlation between the features.

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns

cols=['Cement','BlastFurnaceSlag','FlyAsh','Water','Superplasticizer',
'CoarseAggregate','FineAggregate','Age','ConcreteCompressiveStrength']

data=pd.read_csv('concrete.csv',header=None)
data.columns=cols
data.head()
```

Out[ ]:

	Cement	BlastFurnaceSlag	FlyAsh	Water	Superplasticizer	CoarseAggregate
0	252.000000	0.0	0.000000	185.000000	0.0	1111.000000
1	295.799988	0.0	0.000000	185.699997	0.0	1091.400024
2	252.300003	0.0	98.800003	146.300003	14.2	987.799988
3	172.399994	13.6	172.399994	156.800003	4.1	1006.299988
4	162.000000	214.0	164.000000	202.000000	10.0	820.000000

In [ ]:

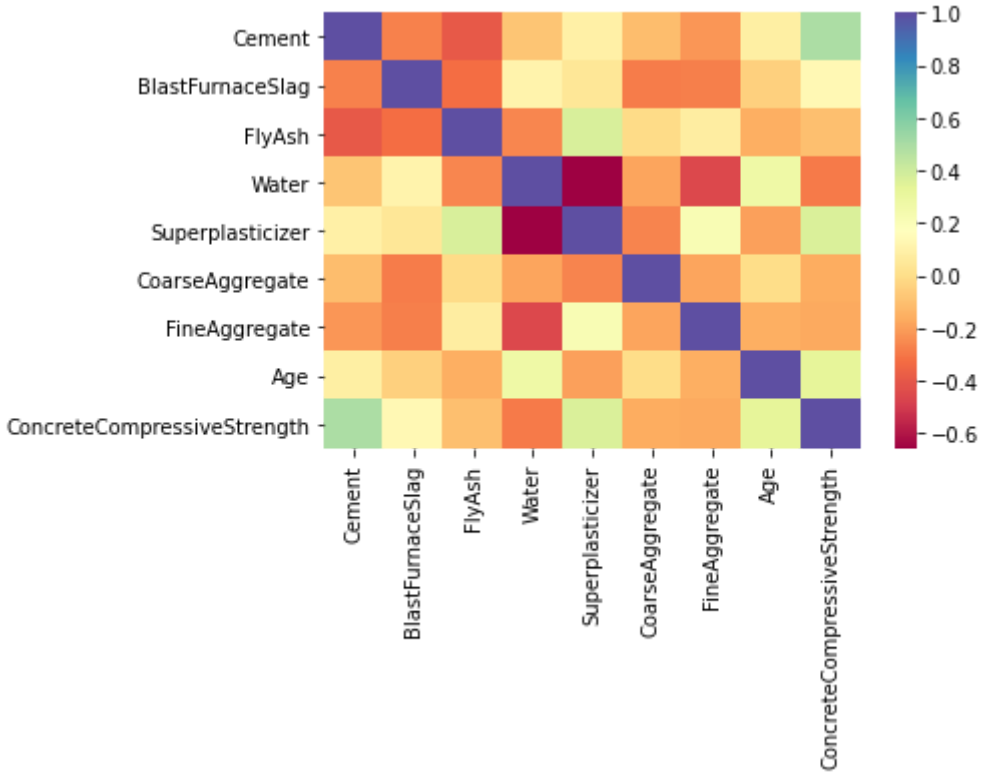
```
data.corr()
```

Out[ ]:

	Cement	BlastFurnaceSlag	FlyAsh	Water	Superplastic
Cement	1.000000	-0.275216	-0.397467	-0.081587	0.092
BlastFurnaceSlag	-0.275216	1.000000	-0.323580	0.107252	0.043
FlyAsh	-0.397467	-0.323580	1.000000	-0.256984	0.377
Water	-0.081587	0.107252	-0.256984	1.000000	-0.657
Superplasticizer	0.092386	0.043270	0.377503	-0.657533	1.000
CoarseAggregate	-0.109349	-0.283999	-0.009961	-0.182294	-0.265
FineAggregate	-0.222718	-0.281603	0.079109	-0.450661	0.222
Age	0.081946	-0.044246	-0.154371	0.277618	-0.192
ConcreteCompressiveStrength	0.497832	0.134829	-0.105755	-0.289633	0.366

In [ ]:

```
sns.heatmap(data.corr(),cmap='Spectral')
plt.show()
```



Question 7 - Intermediate (Bridging Question)

Fit a Linear Regression model by applying 5-fold cross validation using scikit and calculate the scores and average accuracy.

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

X=data.drop('ConcreteCompressiveStrength',axis=1)
y=data.ConcreteCompressiveStrength
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,test_size=0.3,
random_state=101)

lin_model = LinearRegression()

# fitting a model and computing the score 5 consecutive times (with different splits ea
ch time)
scores = cross_val_score(lin_model,X ,y , cv=5)

print("Scores: %s"%scores)
print("Accuracy:  %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Scores: [0.67703939 0.64538043 0.56344555 0.6113636  0.51208946]
Accuracy:  0.60 (+/- 0.12)
```

Question 8 - Advanced (Bridging Question)

Fit a Linear Regression model by applying 5-fold cross validation explicitly using KFold scikit and calculate the scores using mean absolute error as scoring parameter.

In [ ]:

```
from sklearn.model_selection import KFold
k_fold  = KFold(n_splits=5, random_state=101)
scores = cross_val_score(lin_model,X, y, cv=k_fold, scoring='neg_mean_absolute_error')

print("Scores:",scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (-scores.mean(), scores.std() * 2))
```

```
Scores: [-8.06090082 -8.17048162 -8.23717288 -7.88721051 -9.06507441]
Accuracy: 8.28 (+/- 0.82)
```

Question 9 - Advanced

Using LOOCV, fit a Linear Regression model and calculate the average score using mean absolute error as scoring parameter.

In [ ]:

```
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import mean_absolute_error

# creating the leav one out function
loo = LeaveOneOut()
loo.get_n_splits(X)
scores=[]
# printing the training and validation data
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index,:], X.iloc[test_index,:]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    model=LinearRegression()
    model.fit(X_train,y_train)
    scores.append(mean_absolute_error(y_test,model.predict(X_test)))

print('Average Score: %.2f'%np.mean(scores))
```

```
Average Score: 8.30
```

Question 10 - Beginner

Apply Lasso Regression over the model and find r2_score and MAE.

In [ ]:

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error, r2_score

X.shape,y.shape
```

Out[ ]:

```
((1030, 8), (1030,))
```

In [ ]:

```
lasso=Lasso(alpha=1)
lasso.fit(X,y)
```

Out[ ]:

```
Lasso(alpha=1, copy_X=True, fit_intercept=True, max_iter=1000, normalize=F
alse,
      positive=False, precompute=False, random_state=None, selection='cycl
ic',
      tol=0.0001, warm_start=False)
```

In [ ]:

```
print('coeficients: %(coef)s, intercept: %(intercept).2f'%{'coef':lasso.coef_,'intercep
t':lasso.intercept_})
```

```
coeficients: [ 0.11832317  0.10194315  0.08691391 -0.16755123  0.22482698
0.01430917
  0.01737661  0.11375137], intercept: -13.22
```

In [ ]:

```
print('Mean Squared Error: %.2f'%mean_absolute_error(y,lasso.predict(X)))
print('R2-Score: %.2f'%r2_score(y, lasso.predict(X)))
```

```
Mean Squared Error: 8.22
R2-Score: 0.62
```

Question 11 - Beginner

Apply Ridge Regression over the model and find r2_score and MAE.

In [ ]:

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error, r2_score

X.shape,y.shape
```

Out[ ]:

```
((1030, 8), (1030,))
```

In [ ]:

```
ridge=Lasso(alpha=10)
ridge.fit(X,y)
```

Out[ ]:

```
Lasso(alpha=10, copy_X=True, fit_intercept=True, max_iter=1000, normalize=
False,
      positive=False, precompute=False, random_state=None, selection='cycl
ic',
      tol=0.0001, warm_start=False)
```

In [ ]:

```
print('coeficients: %(coef)s, intercept: %(intercept).2f'%{'coef':ridge.coef_,'intercep
t':ridge.intercept_})
```

```
coeficients: [ 0.11027374  0.09109642  0.07846055 -0.22035185  0.
0.
  0.00521729  0.10900354], intercept: 24.82
```

In [ ]:

```
print('Mean Squared Error: %.2f'%mean_absolute_error(y,ridge.predict(X)))
print('R2-Score: %.2f'%r2_score(y, ridge.predict(X)))
```

```
Mean Squared Error: 8.29
R2-Score: 0.61
```

Thus, we can conclude that Lasso Regression has R2 Score of 0.62 which is better than that of Ridge Regression.

Questions 12-16:

Scenario 3 - Toronto Rookies

The National Basketball Association (NBA) is a men's professional basketball league in North America, composed of 30 teams. It is one of the four major sports leagues in USA and Canada. The NBA is an active member of USA Basketball (USAB), which is recognized by FIBA (also known as the International Basketball Federation) as the national governing body for basketball in the United States.

Problem Statement:

The data contains the details of NBA rookies who have recently been part of the NBA. We have to classify the career longevity of NBA rookies more than 5 years or not.

Dataset Description:

NBA.csv - The dataset has details of NBA rookies along with their career longevity either more than 5 years or not:

Attributes:

- **Name**: Name of the players
- **GP**: Games played
- **MIN**: Total minutes played
- **PTS**: Points per game
- **FGM**: Field goals made
- **FGA**: Field goals attempted
- **FG%**: Field goals percent
- **3P Made**: 3 Points made
- **3PA**: 3 Points attempted
- **3P%**: 3 Points percentage
- **FTM**: Free throw made
- **FTA**: Free throw attempted
- **FT%**: Free throw percentage
- **OREB**: Offensive rebounds
- **DREB**: Defensive rebounds
- **REB**: Rebounds
- **AST**: Assists
- **STL**: Steals
- **BLK**: Blocks
- **TOV**: Turnovers

Target Variable:

- **TARGET_5Yrs**: 0 if career years played < 5 | 1 if career years played >= 5

Tasks to be performed:

To attain the above goal, the below tasks must be performed:

- Read the dataset and preprocess the data - **Beginner**

- Split the data into training and testing set; Apply Gaussian Naive Bayes and Decision Tree over the model - **Intermediate**
- Evaluate the models using Confusion Matrix - **Intermediate**
- Apply XGBoost Classifier over the data and evaluate the model - **Advanced**
- Apply GridSearchCV over XGboost Classifier **Advanced**

Topics covered:

- XGBoost
- GridSearchCV
- Decision Tree
- Naive Bayes

Question 12 - Beginner

Read the dataset and preprocess the data.

In [ ]:

```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

data=pd.read_csv('NBA.csv')
data.head()
```

Out[ ]:

	Name	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	FTM	FTA	FT%	OREB
0	Brandon Ingram	36	27.4	7.4	2.6	7.6	34.7	0.5	2.1	25.0	1.6	2.3	69.9	0.7
1	Andrew Harrison	35	26.9	7.2	2.0	6.7	29.6	0.7	2.8	23.5	2.6	3.4	76.5	0.5
2	JaKarr Sampson	74	15.3	5.2	2.0	4.7	42.2	0.4	1.7	24.4	0.9	1.3	67.0	0.5
3	Malik Sealy	58	11.6	5.7	2.3	5.5	42.6	0.1	0.5	22.6	0.9	1.3	68.9	1.0
4	Matt Geiger	48	11.5	4.5	1.6	3.0	52.4	0.0	0.1	0.0	1.3	1.9	67.4	1.0

In [ ]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Name         1340 non-null   object 
 1   GP           1340 non-null   int64  
 2   MIN          1340 non-null   float64
 3   PTS          1340 non-null   float64
 4   FGM          1340 non-null   float64
 5   FGA          1340 non-null   float64
 6   FG%          1340 non-null   float64
 7   3P Made      1340 non-null   float64
 8   3PA          1340 non-null   float64
 9   3P%          1329 non-null   float64
 10  FTM          1340 non-null   float64
 11  FTA          1340 non-null   float64
 12  FT%          1340 non-null   float64
 13  OREB         1340 non-null   float64
 14  DREB         1340 non-null   float64
 15  REB          1340 non-null   float64
 16  AST          1340 non-null   float64
 17  STL          1340 non-null   float64
 18  BLK          1340 non-null   float64
 19  TOV          1340 non-null   float64
 20  TARGET_5Yrs  1340 non-null   float64
dtypes: float64(19), int64(1), object(1)
memory usage: 220.0+ KB
```

In [ ]:

```
data.describe(include='all')
```

Out[ ]:

	Name	GP	MIN	PTS	FGM	FGA	FG
count	1340	1340.000000	1340.000000	1340.000000	1340.000000	1340.000000	1340.000(
unique	1294	NaN	NaN	NaN	NaN	NaN	N
top	Charles Smith	NaN	NaN	NaN	NaN	NaN	N
freq	9	NaN	NaN	NaN	NaN	NaN	N
mean	NaN	60.414179	17.624627	6.801493	2.629104	5.885299	44.169
std	NaN	17.433992	8.307964	4.357545	1.683555	3.593488	6.137(
min	NaN	11.000000	3.100000	0.700000	0.300000	0.800000	23.800(
25%	NaN	47.000000	10.875000	3.700000	1.400000	3.300000	40.200(
50%	NaN	63.000000	16.100000	5.550000	2.100000	4.800000	44.100(
75%	NaN	77.000000	22.900000	8.800000	3.400000	7.500000	47.900(
max	NaN	82.000000	40.900000	28.200000	10.200000	19.800000	73.700(

In [ ]:

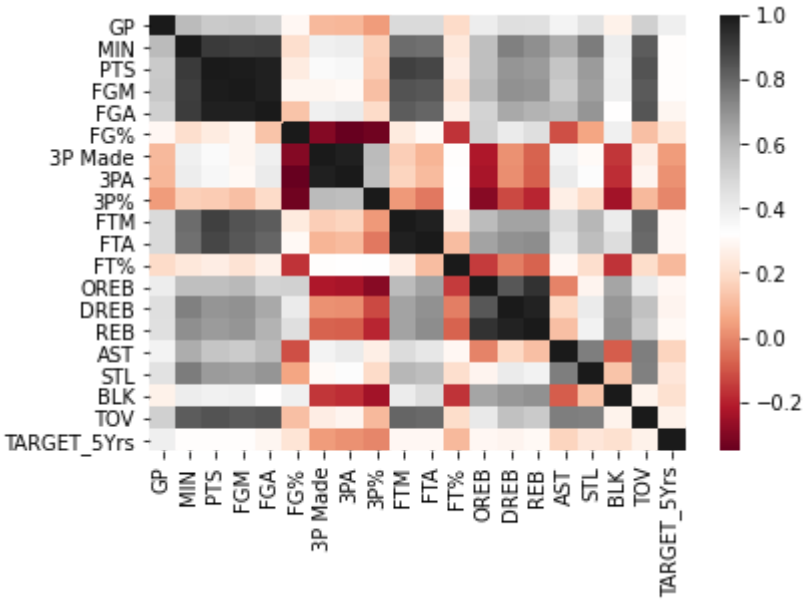
```
data.corr()
```

Out[ ]:

	GP	MIN	PTS	FGM	FGA	FG%	3P Made	
GP	1.000000	0.590240	0.538471	0.542724	0.516625	0.296289	0.107423	0.098
MIN	0.590240	1.000000	0.911822	0.903060	0.910247	0.203901	0.389920	0.403
PTS	0.538471	0.911822	1.000000	0.990834	0.979733	0.255333	0.346682	0.356
FGM	0.542724	0.903060	0.990834	1.000000	0.980050	0.291693	0.289007	0.299
FGA	0.516625	0.910247	0.979733	0.980050	1.000000	0.129798	0.390253	0.413
FG%	0.296289	0.203901	0.255333	0.291693	0.129798	1.000000	-0.294471	-0.350
3P Made	0.107423	0.389920	0.346682	0.289007	0.390253	-0.294471	1.000000	0.982
3PA	0.098772	0.403258	0.356751	0.299057	0.413560	-0.350658	0.982616	1.000
3P%	0.038209	0.165997	0.151072	0.119493	0.197160	-0.330690	0.589855	0.582
FTM	0.482123	0.791000	0.896297	0.848019	0.826616	0.245776	0.158472	0.173
FTA	0.479487	0.779609	0.880703	0.840408	0.805559	0.300154	0.095396	0.108
FT%	0.196299	0.239878	0.258931	0.223566	0.269614	-0.161183	0.314355	0.323
OREB	0.401136	0.573062	0.575106	0.596687	0.504212	0.511367	-0.219010	-0.231
DREB	0.466840	0.745513	0.693934	0.703278	0.640123	0.410555	0.016570	0.011
REB	0.460406	0.709707	0.676849	0.691186	0.614328	0.465423	-0.072503	-0.080
AST	0.372749	0.629015	0.552338	0.532534	0.589818	-0.108797	0.376604	0.410
STL	0.451137	0.757034	0.675341	0.662640	0.690168	0.056658	0.306908	0.338
BLK	0.276498	0.399088	0.387043	0.398125	0.322184	0.391626	-0.158535	-0.172
TOV	0.518167	0.826500	0.850366	0.834352	0.845989	0.121806	0.258369	0.283
TARGET_5Yrs	0.396833	0.317805	0.315981	0.317594	0.292660	0.227134	0.036619	0.018

In [ ]:

```
sns.heatmap(data.corr(),cmap='RdGy')
plt.show()
```



In [ ]:

```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                        [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                        'Percentage':[sum(data[y].isnull())/data.shape[0] for y in dat
a.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
9	3P%	True	11	0.008209
0	Name	False	0	0.000000
11	FTA	False	0	0.000000
19	TOV	False	0	0.000000
18	BLK	False	0	0.000000
17	STL	False	0	0.000000
16	AST	False	0	0.000000
15	REB	False	0	0.000000
14	DREB	False	0	0.000000
13	OREB	False	0	0.000000
12	FT%	False	0	0.000000
10	FTM	False	0	0.000000
1	GP	False	0	0.000000
8	3PA	False	0	0.000000
7	3P Made	False	0	0.000000
6	FG%	False	0	0.000000
5	FGA	False	0	0.000000
4	FGM	False	0	0.000000
3	PTS	False	0	0.000000
2	MIN	False	0	0.000000
20	TARGET_5Yrs	False	0	0.000000

In [ ]:

```
data['3P%'].fillna(data['3P%'].mean(),inplace=True)

miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                           [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in dat
a.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	Name	False	0	0.0
11	FTA	False	0	0.0
19	TOV	False	0	0.0
18	BLK	False	0	0.0
17	STL	False	0	0.0
16	AST	False	0	0.0
15	REB	False	0	0.0
14	DREB	False	0	0.0
13	OREB	False	0	0.0
12	FT%	False	0	0.0
10	FTM	False	0	0.0
1	GP	False	0	0.0
9	3P%	False	0	0.0
8	3PA	False	0	0.0
7	3P Made	False	0	0.0
6	FG%	False	0	0.0
5	FGA	False	0	0.0
4	FGM	False	0	0.0
3	PTS	False	0	0.0
2	MIN	False	0	0.0
20	TARGET_5Yrs	False	0	0.0

In [ ]:

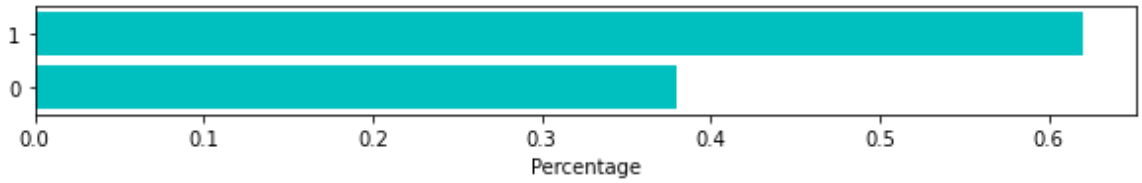
```
target_ratio=pd.DataFrame({'Counts':data.TARGET_5Yrs.value_counts(),'Percentage':data.T
ARGET_5Yrs.value_counts()/len(data)})
target_ratio
```

Out[ ]:

	Counts	Percentage
1.0	831	0.620149
0.0	509	0.379851

In [ ]:

```
plt.figure(figsize = (10,1))
plt.barh(target_ratio.index, target_ratio.Percentage,color='c')
plt.xlabel('Percentage')
plt.show()
```



Question 13 - Intermediate (Bridging Question)

Split the data into training and testing set; Apply Gaussian Naive Bayes and Decision Tree over the model.

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier

X=data.drop(['Name','TARGET_5Yrs'],axis=1)
y=data.TARGET_5Yrs
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,test_size=0.3,
random_state=101)

gnb=GaussianNB()
gnb.fit(X_train,y_train)
g_pred=gnb.predict(X_test)

dnb=DecisionTreeClassifier()
dnb.fit(X_train,y_train)
d_pred=dnb.predict(X_test)
```

Question 14 - Intermediate

Evaluate the models using Confusion Matrix.

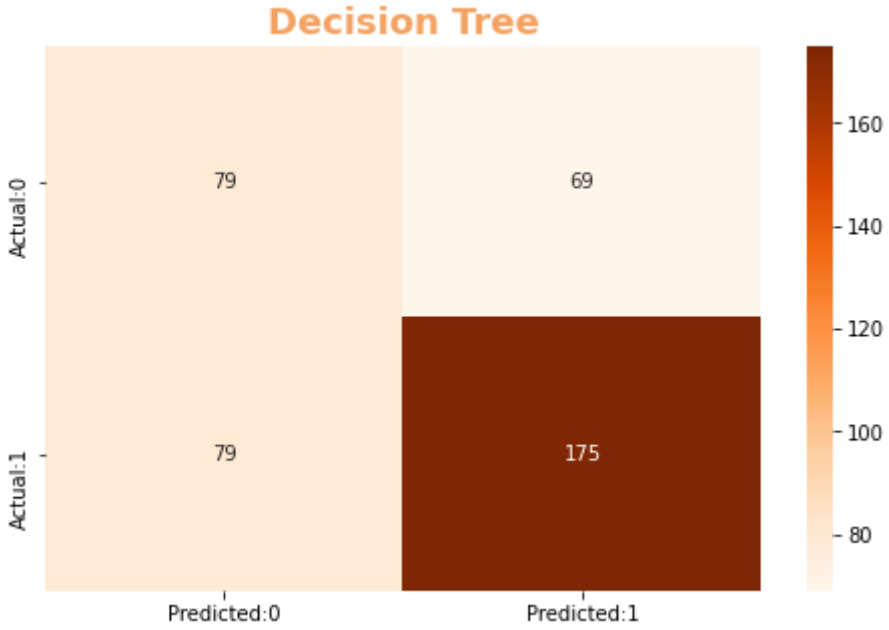
In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix

print('The accuracy of the Gaussian Naive Bayes model is: ',round(accuracy_score(y_test
,g_pred)*100,2))
print('The accuracy of the Decision Tree model is: ',round(accuracy_score(y_test,d_pred
)*100,2))
```

```
The accuracy of the Gaussian Naive Bayes model is:  62.69
The accuracy of the Decision Tree model is:  63.18
```

In [ ]:

```
cm=confusion_matrix(y_test,d_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:
0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap='Oranges')
fm={'size':18,'color':'#f4a261','weight':'bold'}
plt.title('Decision Tree',**fm)
plt.show()
```



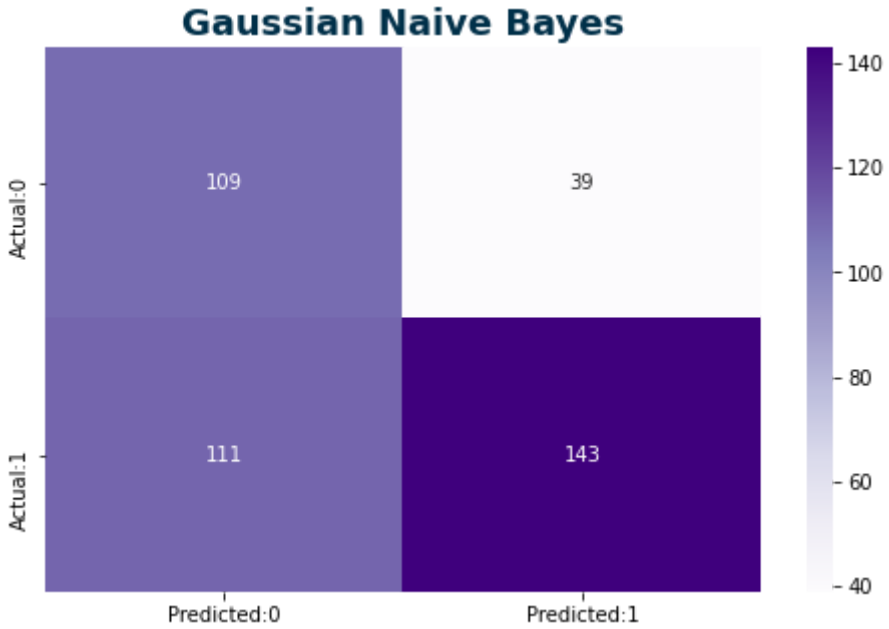
In [ ]:

```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 79
True Positive 175
False Negative 79
False Positive 69
Sensitivity 0.6889763779527559
Specificity 0.5337837837837838
```

In [ ]:

```
cm=confusion_matrix(y_test,g_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:
0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap='Purples')
fm={'size':18,'color':'#003049','weight':'bold'}
plt.title('Gaussian Naive Bayes',**fm)
plt.show()
```



In [ ]:

```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 109
True Positive 143
False Negative 111
False Positive 39
Sensitivity 0.562992125984252
Specificity 0.7364864864864865
```

Thus, we can see that Gaussion Model is more specific than Decision Tree.

Question 15 - Advanced

Apply XGBoost Classifier over the data and evaluate the model.

In [ ]:

```
from xgboost import XGBClassifier

xb_clf=XGBClassifier(learning_rate=0.65,n_estimators=1000)
xb_clf.fit(X_train,y_train)
xb_pred=xb_clf.predict(X_test)

print('The accuracy of the XGBoost model is: ',round(accuracy_score(y_test,xb_pred)*100
,2))
```

```
The accuracy of the XGBoost model is:  70.15
```

Question 16 - Advanced

Apply GridSearchCV over XGboost Classifier.

In [ ]:

```
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")

params={'learning_rate':[0.65,0.70,0.75],'n_estimators':[100,1000]}
rf_gsv=GridSearchCV(estimator=XGBClassifier(),param_grid=params,cv=3,scoring='accuracy'
)
rf_gsv.fit(X,y)
```

Out[ ]:

```
GridSearchCV(cv=3, error_score=nan,
             estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                     colsample_bylevel=1, colsample_bynode
=1,
                                     colsample_bytree=1, gamma=0,
                                     learning_rate=0.1, max_delta_step=0,
                                     max_depth=3, min_child_weight=1,
                                     missing=None, n_estimators=100, n_job
s=1,
                                     nthread=None, objective='binary:logis
tic',
                                     random_state=0, reg_alpha=0, reg_lamb
da=1,
                                     scale_pos_weight=1, seed=None, silent
=None,
                                     subsample=1, verbosity=1),
             iid='deprecated', n_jobs=None,
             param_grid={'learning_rate': [0.65, 0.7, 0.75],
                         'n_estimators': [100, 1000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring='accuracy', verbose=0)
```

Here, we have performed XGBoost hyperparameter tuning by doing a grid search.

In [ ]:

```
pd.DataFrame(rf_gsv.cv_results_).sort_values('rank_test_score')
```

Out[ ]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_learning_rate	param
5	0.827096	0.021178	0.017588	0.000955	0.75	
2	0.090022	0.000283	0.003177	0.000017	0.7	
3	0.859241	0.026631	0.018861	0.002116	0.7	
1	0.856329	0.025052	0.017166	0.000534	0.65	
4	0.092827	0.000306	0.003642	0.000042	0.75	
0	0.093323	0.003231	0.003313	0.000020	0.65	

In [ ]:

```
rf_gsv.best_params_
```

Out[ ]:

```
{'learning_rate': 0.75, 'n_estimators': 1000}
```

In [ ]:

```
rf_gsv.best_estimator_
```

Out[ ]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.75, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=
1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Questions 17-21:

Scenario 4 - Shurima Surveys

Shurima Surveys is a non-government organization whose mission is to empower people and communities in state of poverty, illiteracy, disease, and social injustice. But the organization specializes in suicide preventions and mental healthcare. Surveys were conducted on people living in different rural zones. They record based on there lifestyle, assets, income and so on.

Problem Statement:

The aim is to classify if the individual is depressed or not based on the data recorded.

Dataset Description:

Depressed.csv - The dataset contains various details of individuals along with their emotional status i.e. whether they are depressed or not. The details recorded during the survey were:

Attributes:

- **Survey_id**: ID of the survey conducted
- **Ville_id**: Village ID where the survey was conducted(1-292)
- **sex**: Male or Female
- **Age**: Age of the individual(17-91)
- **Married**: Whether the individual is married or not
- **Number_children**: Number of children the individual has(0-11)
- **education_level**: Education level(1-19)
- **total_members**: Total number of members in the family(1-12)
- **gained_asset**: Total assets gained(325K-99.1M)
- **durable_asset**: Total durable assets gained(163K-99.6M)
- **save_asset**: Assets saved(173K-99.9M)
- **living_expenses**: Expense of living(263K-99.3M)
- **other_expenses**: Other expenses(173K-99.8M)
- **Salaried**: Salaried or not
- **incoming_own_farm**: Does individual have a farm or not
- **incoming_business**: Any Business
- **incoming_agricultural**: Agricultural income(325K-99.8M)
- **farm_expenses**: Farming expenses(272K-99.7K)
- **lasting_investment**: Lasting investments(74.3K-99.4M)
- **no_lasting_investmen**: Non-Lasting investments(126K-99.6M)

Target Variable:

- **depressed**: Whether the individual is depressed or not

Tasks to be performed:

To attain the above goal, the below tasks must be performed:

- Read the data and remove all the missing values - **Beginner**
- Preprocess the data using LabelEncoder and split into training and testing set - **Beginner**
- Apply Gradient Boosting Classifier and evaluate the model -**Intermediate**

- Apply AdaBoost Classifier and evaluate the model - **Intermediate**
- Find the ROC-AUC Score of the models and plot on ROC-AUC Curve - **Advanced**

Topics covered:

- ROC-AUC
- AdaBoost
- GradientBoost

Question 17 - Beginner

Read the data and remove all the missing values.

In [ ]:

```
import pandas as pd
import numpy as np

data=pd.read_csv('Depressed.csv')
data.head()
```

Out[ ]:

	Survey_id	Ville_id	sex	Age	Married	Number_children	education_level	total_membe
0	926	91	Male	28	Married	4	10	
1	747	57	Male	23	Married	3	8	
2	1190	115	Male	22	Married	3	9	
3	1065	97	Male	27	Married	2	10	
4	806	42	Female	59	Not Married	4	10	

In [ ]:

```
data.shape
```

Out[ ]:

```
(1429, 21)
```

In [ ]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1429 entries, 0 to 1428
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Survey_id              1429 non-null   int64  
 1   Ville_id               1429 non-null   int64  
 2   sex                    1429 non-null   object 
 3   Age                    1429 non-null   int64  
 4   Married                1429 non-null   object 
 5   Number_children        1429 non-null   int64  
 6   education_level        1429 non-null   int64  
 7   total_members          1429 non-null   int64  
 8   gained_asset           1429 non-null   int64  
 9   durable_asset          1429 non-null   int64  
 10  save_asset             1429 non-null   int64  
 11  living_expenses        1429 non-null   int64  
 12  other_expenses         1429 non-null   int64  
 13  Salaried               1429 non-null   object 
 14  incoming_own_farm      1429 non-null   object 
 15  incoming_business      1429 non-null   object 
 16  incoming_agricultural  1429 non-null   int64  
 17  farm_expenses          1429 non-null   int64  
 18  lasting_investment     1429 non-null   int64  
 19  no_lasting_investmen   1409 non-null   float64
 20  depressed              1429 non-null   object 
dtypes: float64(1), int64(14), object(6)
memory usage: 234.6+ KB
```

In [ ]:

```
data.describe(include='all')
```

Out[ ]:

	Survey_id	Ville_id	sex	Age	Married	Number_children	education_le
count	1429.00000	1429.000000	1429	1429.000000	1429	1429.000000	1429.000(
unique	NaN	NaN	2	NaN	2	NaN	N
top	NaN	NaN	Male	NaN	Married	NaN	N
freq	NaN	NaN	1312	NaN	1104	NaN	N
mean	715.00000	76.286214	NaN	34.777467	NaN	2.883135	8.6871
std	412.66108	66.444012	NaN	13.986219	NaN	1.874472	2.923(
min	1.00000	1.000000	NaN	17.000000	NaN	0.000000	1.000(
25%	358.00000	24.000000	NaN	25.000000	NaN	2.000000	8.000(
50%	715.00000	57.000000	NaN	30.000000	NaN	3.000000	9.000(
75%	1072.00000	105.000000	NaN	42.000000	NaN	4.000000	10.000(
max	1429.00000	292.000000	NaN	91.000000	NaN	11.000000	19.000(

In [ ]:

```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                        [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                        'Percentage':[sum(data[y].isnull())/data.shape[0] for y in dat
a.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
19	no_lasting_investmen	True	20	0.013996
0	Survey_id	False	0	0.000000
11	living_expenses	False	0	0.000000
18	lasting_investment	False	0	0.000000
17	farm_expenses	False	0	0.000000
16	incoming_agricultural	False	0	0.000000
15	incoming_business	False	0	0.000000
14	incoming_own_farm	False	0	0.000000
13	Salaried	False	0	0.000000
12	other_expenses	False	0	0.000000
10	save_asset	False	0	0.000000
1	Ville_id	False	0	0.000000
9	durable_asset	False	0	0.000000
8	gained_asset	False	0	0.000000
7	total_members	False	0	0.000000
6	education_level	False	0	0.000000
5	Number_children	False	0	0.000000
4	Married	False	0	0.000000
3	Age	False	0	0.000000
2	sex	False	0	0.000000
20	depressed	False	0	0.000000

In [ ]:

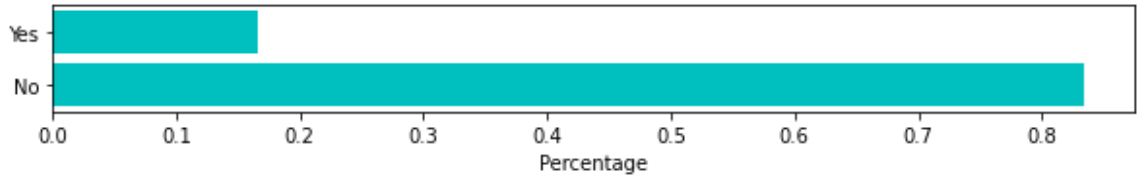
```
data.no_lasting_investmen.fillna(data.no_lasting_investmen.mean(),inplace=True)

miss=pd.DataFrame({'Col_name':data.columns,'Missing value?':
                        [any(data[x].isnull()) for x in data.columns],
                   'Count_':[sum(data[y].isnull()) for y in data.columns],
                           'Percentage':[sum(data[y].isnull())/data.shape[0] for y in dat
a.columns]})
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_	Percentage
0	Survey_id	False	0	0.0
11	living_expenses	False	0	0.0
19	no_lasting_investmen	False	0	0.0
18	lasting_investment	False	0	0.0
17	farm_expenses	False	0	0.0
16	incoming_agricultural	False	0	0.0
15	incoming_business	False	0	0.0
14	incoming_own_farm	False	0	0.0
13	Salaried	False	0	0.0
12	other_expenses	False	0	0.0
10	save_asset	False	0	0.0
1	Ville_id	False	0	0.0
9	durable_asset	False	0	0.0
8	gained_asset	False	0	0.0
7	total_members	False	0	0.0
6	education_level	False	0	0.0
5	Number_children	False	0	0.0
4	Married	False	0	0.0
3	Age	False	0	0.0
2	sex	False	0	0.0
20	depressed	False	0	0.0

In [ ]:

```
#Check distribution of target variable
target_ratio=pd.DataFrame({'Counts':data.depressed.value_counts(),'Percentage':data.dep
ressed.value_counts()/len(data)})
target_ratio

plt.figure(figsize = (10,1))
plt.barh(target_ratio.index, target_ratio.Percentage,color='c')
plt.xlabel('Percentage')
plt.show()
```



Question 18 - Beginner

Preprocess the data using LabelEncoder and split into training and testing set.

In [ ]:

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

lb=LabelEncoder()
data.depressed=lb.fit_transform(data.depressed)

lb.classes_
```

Out[ ]:

```
array(['No', 'Yes'], dtype=object)
```

In [ ]:

```
scale_cols=['Age','gained_asset', 'durable_asset', 'save_asset',
       'living_expenses', 'other_expenses', 'incoming_agricultural',
       'farm_expenses', 'lasting_investment', 'no_lasting_investmen',]
data[scale_cols]=MinMaxScaler().fit_transform(data[scale_cols])

cols=['sex','Married','Salaried','incoming_own_farm','incoming_business']
data[cols]=data[cols].apply(LabelEncoder().fit_transform)

X=data.drop(['depressed','Survey_id','Ville_id'],axis=1)
y=data.depressed

X_train, X_test, y_train, y_test=train_test_split(X,y,random_state=2,test_size=0.3)
```

Question 19 - Intermediate

Apply Gradient Boosting Classifier and evaluate the model.

In [ ]:

```
from sklearn.metrics import accuracy_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix

gb_clf=GradientBoostingClassifier(learning_rate=0.5,n_estimators=1000)
gb_clf.fit(X_train,y_train)
gb_pred=gb_clf.predict(X_test)
```

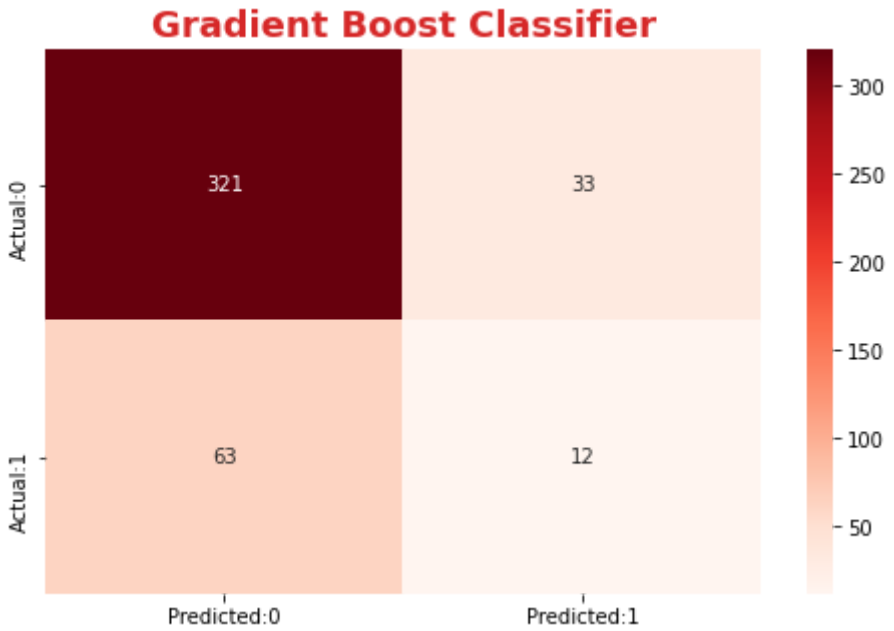
In [ ]:

```
print('The accuracy of the GradientBoostingClassifier model is: ',round(accuracy_score(
y_test,gb_pred)*100,2))
```

```
The accuracy of the GradientBoostingClassifier model is:  77.62
```

In [ ]:

```
cm=confusion_matrix(y_test,gb_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:
0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap='Reds')
fm={'size':18,'color':'#d62828','weight':'bold'}
plt.title('Gradient Boost Classifier',**fm)
plt.show()
```



In [ ]:

```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 321
True Positive 12
False Negative 63
False Positive 33
Sensitivity 0.16
Specificity 0.9067796610169492
```

The model is highly specific.

Question 20 - Intermediate

Apply AdaBoost Classifier and evaluate the model.

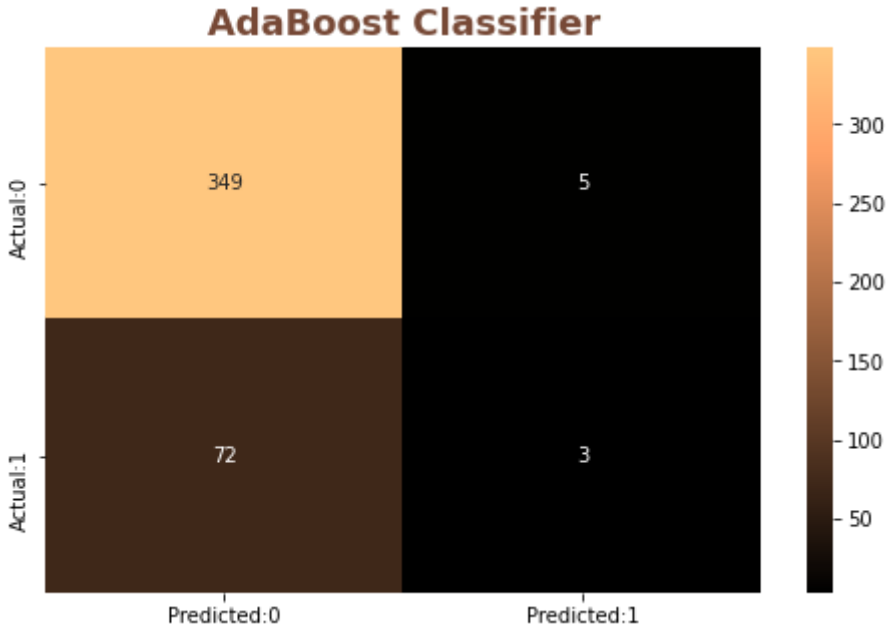
In [ ]:

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix

ada=AdaBoostClassifier(n_estimators=100,learning_rate=0.6)
ada.fit(X_train,y_train)
ada_pred=ada.predict(X_test)

print('The accuracy of the AdaBoost Classifier model is: ',round(accuracy_score(y_test,
ada_pred)*100,2))
```

```
The accuracy of the AdaBoost Classifier model is:  82.05
```

In [ ]:

```
cm=confusion_matrix(y_test,ada_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:
0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap='copper')
fm={'size':18,'color':'#774936','weight':'bold'}
plt.title('AdaBoost Classifier',**fm)
plt.show()
```



In [ ]:

```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print("True Negative", TN)
print("True Positive", TP)
print("False Negative", FN)
print("False Positive", FP)
print("Sensitivity", sensitivity)
print("Specificity", specificity)
```

```
True Negative 349
True Positive 3
False Negative 72
False Positive 5
Sensitivity 0.04
Specificity 0.9858757062146892
```

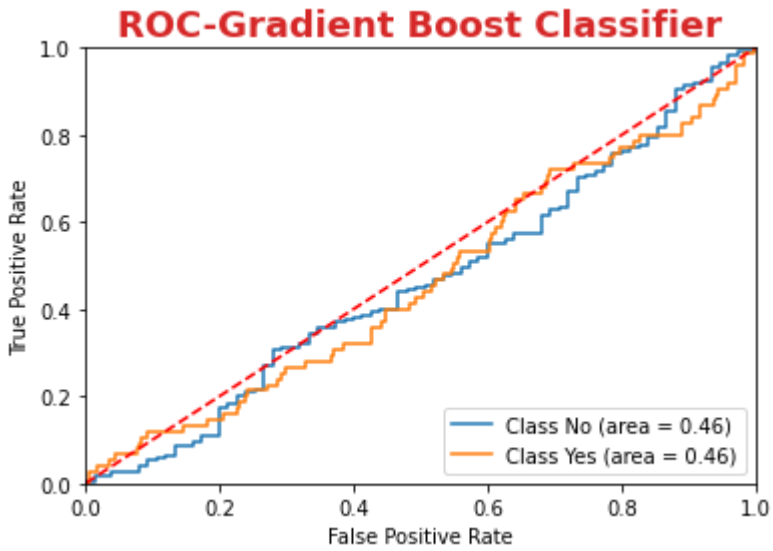
Both the models are highly specific. This is due to the number of positive observations are very less.

Question 21 - Advanced

Find the ROC-AUC Score of the models and plot on ROC-AUC Curve.

In [ ]:

```
from sklearn.metrics import roc_auc_score,roc_curve

adamodel_prob=gb_clf.predict_proba(X_test).T


dummy_y_test=pd.get_dummies(y_test)

roc_auc=dict()
lfpr4=dict()
ltpr6=dict()
lthresholds4=dict()
for i in dummy_y_test.columns:
    roc_auc[i]=roc_auc_score(dummy_y_test[i],adamodel_prob[i-1])
    lfpr4[i], ltpr6[i], lthresholds4[i] = roc_curve(dummy_y_test[i], adamodel_prob[i-1
])


for i in dummy_y_test.columns:
    cls=lb.classes_
    plt.plot(lfpr4[i], ltpr6[i], label='Class '+str(cls[i])+ ' (area = %0.2f)' % roc_au
c[i])



plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
fm={'size':18,'color':'#d62828','weight':'bold'}
plt.title('ROC-Gradient Boost Classifier',**fm)
plt.legend(loc="lower right")
plt.show()
```



In [ ]:

```
adamodel_prob=ada.predict_proba(X_test).T


dummy_y_test=pd.get_dummies(y_test)

roc_auc=dict()
lfpr4=dict()
ltpr6=dict()
lthresholds4=dict()
for i in dummy_y_test.columns:
    roc_auc[i]=roc_auc_score(dummy_y_test[i],adamodel_prob[i-1])
    lfpr4[i], ltpr6[i], lthresholds4[i] = roc_curve(dummy_y_test[i], adamodel_prob[i-1
])


for i in dummy_y_test.columns:
    cls=lb.classes_
    plt.plot(lfpr4[i], ltpr6[i], label='Class '+str(cls[i])+ ' (area = %0.2f)' % roc_au
c[i])



plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
fm={'size':18,'color':'#774936','weight':'bold'}
plt.title('ROC-AdaBoost',**fm)
plt.legend(loc="lower right")
plt.show()
```