# lab1-logisticregression

June 14, 2023

Heart Disease prediction

Splitting data, logistic Regression

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
```

[ ]:
```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/framingham')
```

[ ]:
```
df
```

[ ]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39        4.0              0         0.0     0.0
1        0   46        2.0              0         0.0     0.0
2        1   48        1.0              1        20.0     0.0
3        0   61        3.0              1        30.0     0.0
4        0   46        3.0              1        23.0     0.0
…      …    …        …              …           …      …
4235     0   48        2.0              1        20.0     NaN
4236     0   44        1.0              1        15.0     0.0
4237     0   52        2.0              0         0.0     0.0
4238     1   40        3.0              0         0.0     0.0
4239     0   39        3.0              1        30.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                 …             …        …       …      …     …      …
```

```
4235                 0              0          0     248.0  131.0    72.0   22.00
4236                 0              0          0     210.0  126.5    87.0   19.16
4237                 0              0          0     269.0  133.5    83.0   21.47
4238                 0              1          0     185.0  141.0    98.0   25.60
4239                 0              0          0     196.0  133.0    86.0   20.91

      heartRate  glucose  TenYearCHD
0          80.0     77.0           0
1          95.0     76.0           0
2          75.0     70.0           0
3          65.0    103.0           1
4          85.0     85.0           0
…            …        …          …
4235       84.0     86.0           0
4236       86.0      NaN           0
4237       80.0    107.0           0
4238       67.0     72.0           0
4239       85.0     80.0           0

[4240 rows x 16 columns]
```

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

[ ]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
0     1   39        4.0              0         0.0     0.0                0
1     0   46        2.0              0         0.0     0.0                0
2     1   48        1.0              1        20.0     0.0                0
3     0   61        3.0              1        30.0     0.0                0
4     0   46        3.0              1        23.0     0.0                0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
1           0
2           0
3           1
4           0
```

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

[ ]:
```
(4240, 16)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4240 non-null   int64
 1   age              4240 non-null   int64
 2   education        4135 non-null   float64
 3   currentSmoker    4240 non-null   int64
 4   cigsPerDay       4211 non-null   float64
 5   BPMeds           4187 non-null   float64
 6   prevalentStroke  4240 non-null   int64
 7   prevalentHyp     4240 non-null   int64
 8   diabetes         4240 non-null   int64
 9   totChol          4190 non-null   float64
 10  sysBP            4240 non-null   float64
 11  diaBP            4240 non-null   float64
 12  BMI              4221 non-null   float64
 13  heartRate        4239 non-null   float64
 14  glucose          3852 non-null   float64
 15  TenYearCHD       4240 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

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

[ ]:
```
male                 0
age                  0
education          105
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
```

[ ]:
```
df.dropna(axis = 0, inplace = True) #axis=0(row),axis=1(col)
```

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

[ ]:
```
male               0
age                0
education          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
```

[ ]:
```
df
```

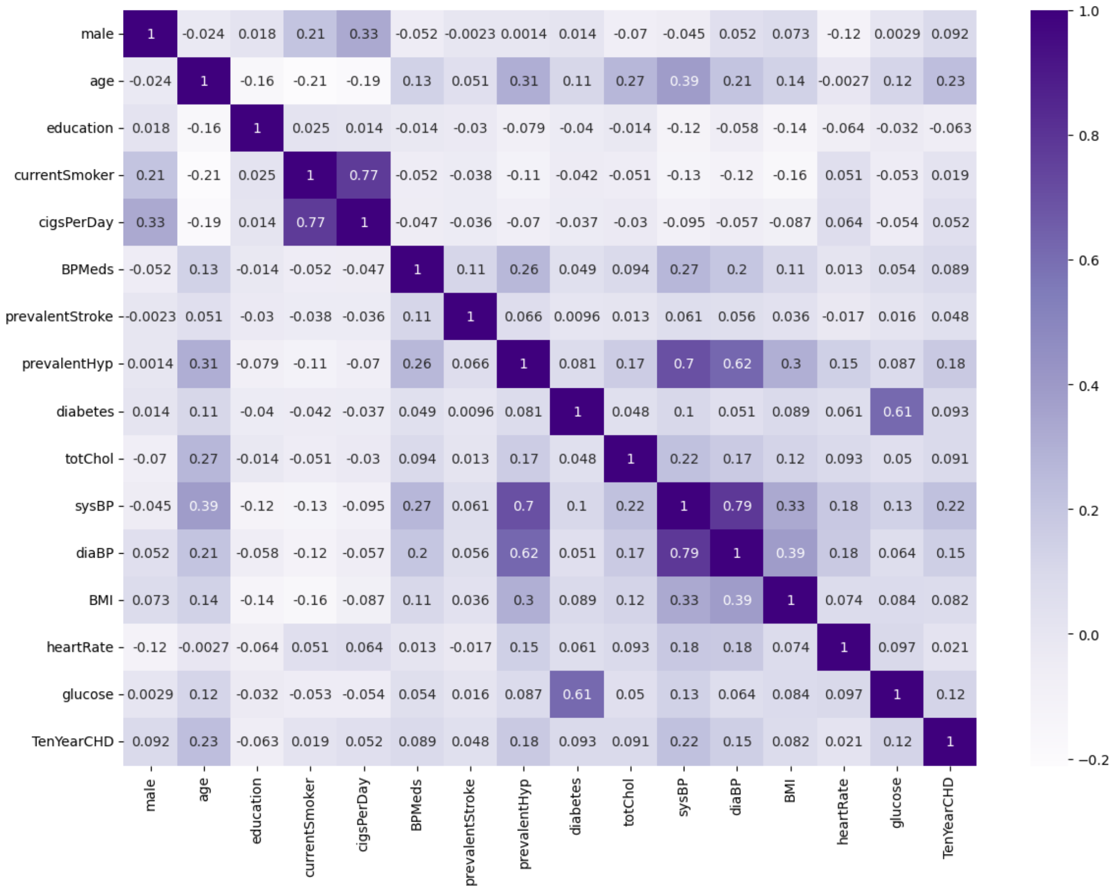
[ ]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39        4.0              0         0.0     0.0
1        0   46        2.0              0         0.0     0.0
2        1   48        1.0              1        20.0     0.0
3        0   61        3.0              1        30.0     0.0
4        0   46        3.0              1        23.0     0.0
…      …    …          …              …           …       …
4233     1   50        1.0              1         1.0     0.0
4234     1   51        3.0              1        43.0     0.0
4237     0   52        2.0              0         0.0     0.0
4238     1   40        3.0              0         0.0     0.0
4239     0   39        3.0              1        30.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                   …             …         …      …      …      …      …
4233                0             1         0    313.0  179.0   92.0  25.97
4234                0             0         0    207.0  126.5   80.0  19.71
4237                0             0         0    269.0  133.5   83.0  21.47
4238                0             1         0    185.0  141.0   98.0  25.60
4239                0             0         0    196.0  133.0   86.0  20.91
```

```
      heartRate  glucose  TenYearCHD
0          80.0     77.0           0
1          95.0     76.0           0
2          75.0     70.0           0
3          65.0    103.0           1
4          85.0     85.0           0
…           …       …           …
4233       66.0     86.0           1
4234       65.0     68.0           0
4237       80.0    107.0           0
4238       67.0     72.0           0
4239       85.0     80.0           0

[3658 rows x 16 columns]
```

[ ]:
```
df['TenYearCHD'].value_counts()  #will count the nos of 0 & 1
```

[ ]:
```
0    3101
1     557
Name: TenYearCHD, dtype: int64
```

[ ]:
```
plt.figure(figsize = (14, 10))
sns.heatmap(df.corr(), annot = True, cmap = 'Purples', linecolor = 'Red')
plt.show()
```




Splitting the data(train,test)

[ ]:
```
x = df.iloc[:,:15]
y = df.iloc[:,15:16]
```

[ ]:
```
x.head()
```

[ ]:

| | male | age | education | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 39 | 4.0 | 0 | 0.0 | 0.0 | 0 | |
| 1 | 0 | 46 | 2.0 | 0 | 0.0 | 0.0 | 0 | |
| 2 | 1 | 48 | 1.0 | 1 | 20.0 | 0.0 | 0 | |
| 3 | 0 | 61 | 3.0 | 1 | 30.0 | 0.0 | 0 | |
| 4 | 0 | 46 | 3.0 | 1 | 23.0 | 0.0 | 0 | |

| | prevalentHyp | diabetes | totChol | sysBP | diaBP | BMI | heartRate | glucose |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 195.0 | 106.0 | 70.0 | 26.97 | 80.0 | 77.0 |
| 1 | 0 | 0 | 250.0 | 121.0 | 81.0 | 28.73 | 95.0 | 76.0 |
| 2 | 0 | 0 | 245.0 | 127.5 | 80.0 | 25.34 | 75.0 | 70.0 |
| 3 | 1 | 0 | 225.0 | 150.0 | 95.0 | 28.58 | 65.0 | 103.0 |

```
4              0          0     285.0   130.0    84.0   23.10         85.0      85.0
```

[ ]:
```
y.head()
```

[ ]:
```
   TenYearCHD
0            0
1            0
2            0
3            1
4            0
```

[ ]:
```
#importing the model and assigning the data to train test data set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
 ↪random_state = 21)
```

[ ]:
```
print(x_train)
print(x_test)
```

```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
2619     0   38        2.0              1        20.0     0.0
774      0   54        2.0              0         0.0     0.0
3710     1   46        3.0              1        30.0     0.0
3718     0   47        2.0              1         5.0     0.0
3138     1   44        1.0              1         9.0     0.0
…     …  …        …            …            …      …
56       0   54        1.0              1         9.0     0.0
886      0   64        1.0              0         0.0     0.0
2162     0   66        1.0              0         0.0     0.0
1425     0   46        2.0              1        20.0     0.0
3507     1   50        2.0              1        16.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
2619                0             0         0    195.0  116.0   72.0  24.45
774                 0             0         0    193.0  118.0   84.0  24.90
3710                0             1         0    154.0  141.0   90.0  22.76
3718                0             0         0    236.0  128.0   81.0  27.42
3138                0             0         0    273.0  114.0   83.0  27.33
…                …            …        …       …     …     …     …
56                  0             0         1    266.0  114.0   76.0  17.61
886                 0             1         0    194.0  176.0   97.0  33.19
2162                0             1         0    212.0  220.0   96.0  44.71
1425                0             0         0    250.0  115.0   74.0  22.70
3507                0             0         0    214.0  114.0   72.0  22.93

      heartRate  glucose
2619       75.0     90.0
```

```
774       70.0    82.0
3710      65.0    65.0
3718      60.0    93.0
3138      70.0    65.0
…          …       …
56        88.0    55.0
886       68.0    89.0
2162     110.0    95.0
1425     100.0    69.0
3507      66.0    83.0

[2560 rows x 15 columns]
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
3019     0   49        2.0              1        20.0     0.0
3412     0   42        3.0              0         0.0     0.0
1729     1   63        4.0              1        20.0     0.0
2547     0   57        1.0              0         0.0     0.0
2837     1   40        3.0              1        15.0     0.0
…     …   …          …             …           …       …
2879     0   53        3.0              0         0.0     0.0
1162     1   61        1.0              0         0.0     0.0
3819     0   53        2.0              0         0.0     0.0
3551     0   43        4.0              1        30.0     0.0
3410     0   51        4.0              1        10.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
3019                0             0         0    273.0  147.0   89.0  24.26
3412                0             0         0    204.0  108.0   70.5  27.71
1729                0             1         0    248.0  135.0   80.0  23.06
2547                0             1         0    254.0  182.5   97.0  27.38
2837                0             0         0    203.0  123.0   82.0  24.74
…                 …             …       …      …     …     …     …
2879                0             0         0    280.0  135.0   87.0  27.17
1162                0             0         0    214.0  100.0   65.0  30.18
3819                0             1         0    230.0  170.0  113.0  29.55
3551                0             0         0    235.0  128.5   80.0  18.83
3410                0             0         0    240.0  112.0   83.0  24.10

      heartRate  glucose
3019       85.0     62.0
3412       75.0     65.0
1729       78.0    118.0
2547       77.0     72.0
2837       75.0     78.0
…          …       …
2879       80.0     80.0
1162       60.0     66.0
3819      115.0    115.0
```

```
3551       90.0     70.0
3410       75.0     77.0

[1098 rows x 15 columns]
```

[ ]:
```python
print(y_train)
print(y_test)
```

```
      TenYearCHD
2619           0
774            0
3710           0
3718           0
3138           0
…          …
56             0
886            1
2162           0
1425           0
3507           0

[2560 rows x 1 columns]
      TenYearCHD
3019           0
3412           0
1729           1
2547           0
2837           0
…          …
2879           0
1162           0
3819           0
3551           0
3410           0

[1098 rows x 1 columns]
```

Logistic Regression

[ ]:
```python
#Applying the ML model - logistic regression
from sklearn.linear_model import LogisticRegression
logreg =LogisticRegression()
```

[ ]:
```python
#Training the data
logreg.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was
```

```
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (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
  n_iter_i = _check_optimize_result(
```

[ ]: LogisticRegression()

[ ]:
```
#Testing the data
y_pred = logreg.predict(x_test)
```

[ ]:
```
#predicting the score
score = logreg.score(x_test, y_test)
print("Prediction Score is : ", score)
```

```
Prediction Score is :  0.848816029143898
```

# lab2-knn-algo

June 14, 2023

using KNN Algorith to predict if a person will have dibetes or not

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

[ ]:
```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/diabetes_.
↪csv')
```

[ ]:
```
data
```

[ ]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …             …             …      …    …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
```

```
1                         0.351   31         0
2                         0.672   32         1
3                         0.167   21         0
4                         2.288   33         1
..                          …   …        …
763                       0.171   63         0
764                       0.340   27         0
765                       0.245   30         0
766                       0.349   47         1
767                       0.315   23         0

[768 rows x 9 columns]
```

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

[ ]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

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

[ ]:
```
(768, 9)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
```

```
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[ ]:
```
data.isnull().sum()
```

[ ]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[ ]:
```
#replacing with mean of respective column

zero_not_accepted = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI',␣
 ↪'Insulin']

for col in zero_not_accepted:
  data[col] = data[col].replace(0, np.NaN)
  mean = int(data[col].mean(skipna=True))
  data[col] = data[col].replace(np.NaN, mean)
```

[ ]:
```
data
```

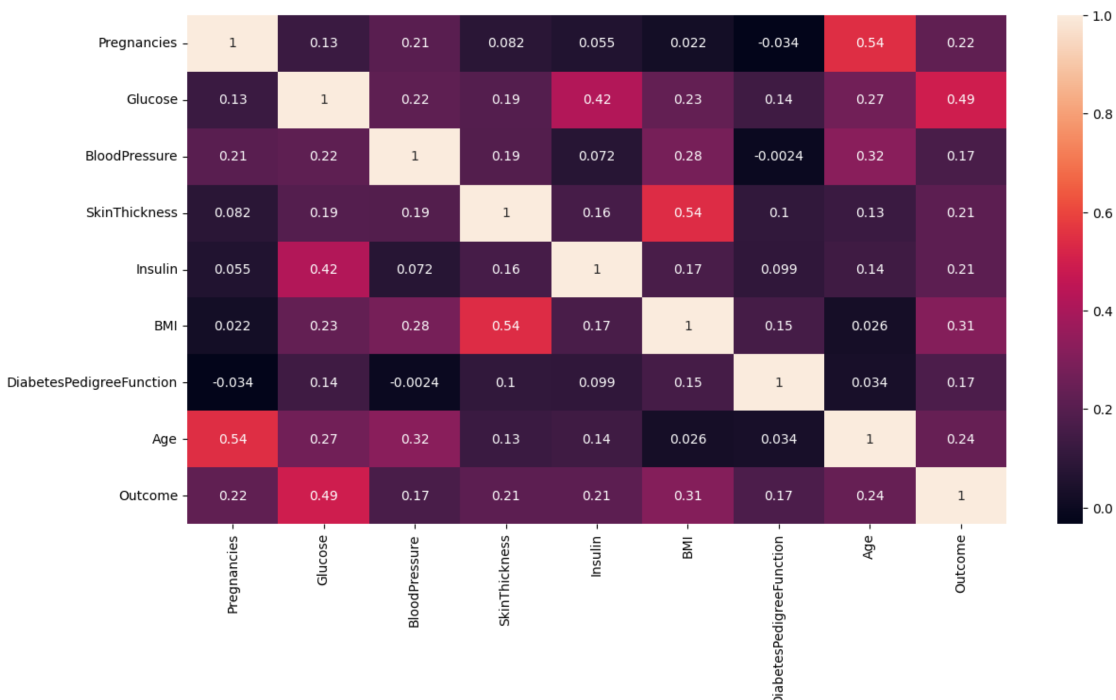
[ ]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6    148.0           72.0           35.0    155.0  33.6
1              1     85.0           66.0           29.0    155.0  26.6
2              8    183.0           64.0           29.0    155.0  23.3
3              1     89.0           66.0           23.0     94.0  28.1
4              0    137.0           40.0           35.0    168.0  43.1
..           …        …             …              …       …     …
763           10    101.0           76.0           48.0    180.0  32.9
764            2    122.0           70.0           27.0    155.0  36.8
765            5    121.0           72.0           23.0    112.0  26.2
766            1    126.0           60.0           29.0    155.0  30.1
767            1     93.0           70.0           31.0    155.0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
```

```
3                        0.167   21         0
4                        2.288   33         1
..                         …   …        …
763                      0.171   63         0
764                      0.340   27         0
765                      0.245   30         0
766                      0.349   47         1
767                      0.315   23         0

[768 rows x 9 columns]
```

[ ]:
```
#Extracting independant variable

x = data.iloc[:,0:8]
```

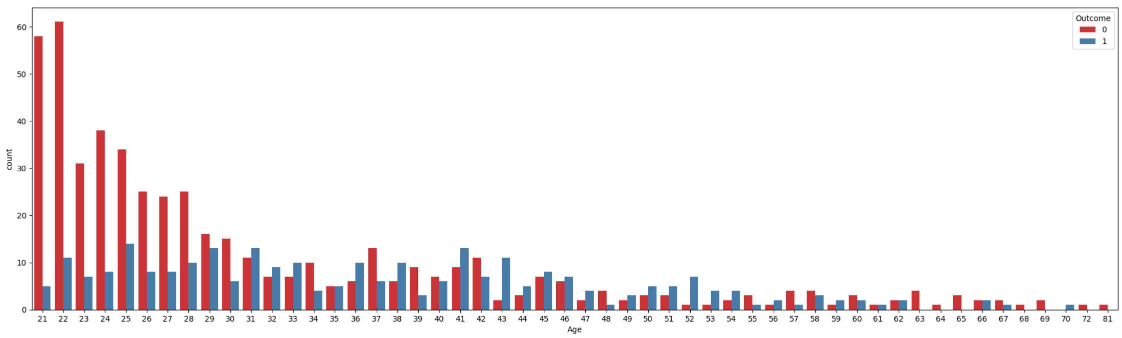
[ ]:
```
#Extracting dependant variable

y = data.iloc[:,8]
```

[ ]:
```
plt.figure(figsize = (14, 7))
sns.heatmap(data.corr(), annot = True)
plt.show()
```



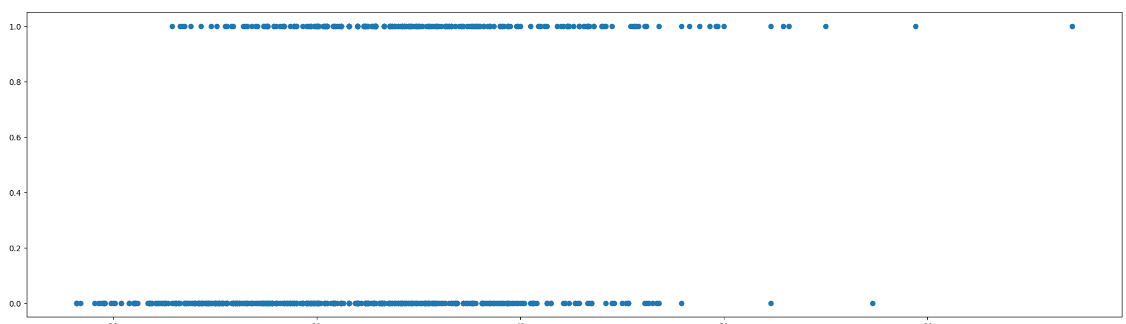
[ ]:
```python
plt.figure(figsize = (25,7))
sns.countplot(x = 'Age', hue = 'Outcome', data = data, palette = 'Set1')
```

[ ]:
```
<Axes: xlabel='Age', ylabel='count'>
```




[ ]:
```python
plt.figure(figsize = (25,7))
plt.scatter(data['BMI'], data['Outcome'])
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7efeecdf9e70>
```



Splitting data

[ ]:
```python
#splitting dataset into training and testing set

x_train, x_test, y_train, y_test = train_test_split(x, y , test_size = 0.2,␣
↪random_state = 0)
```

[ ]:
```python
x_train
```

[ ]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
603            7    150.0           78.0           29.0    126.0  35.2
118            4     97.0           60.0           23.0    155.0  28.2
247            0    165.0           90.0           33.0    680.0  52.3
```

```
157            1    109.0           56.0           21.0    135.0  25.2
468            8    120.0           72.0           29.0    155.0  30.0
..           …      …             …             …      …    …
763           10    101.0           76.0           48.0    180.0  32.9
192            7    159.0           66.0           29.0    155.0  30.4
629            4     94.0           65.0           22.0    155.0  24.7
559           11     85.0           74.0           29.0    155.0  30.1
684            5    136.0           82.0           29.0    155.0  32.0

     DiabetesPedigreeFunction  Age
603                     0.692   54
118                     0.443   22
247                     0.427   23
157                     0.833   23
468                     0.183   38
..                        …   …
763                     0.171   63
192                     0.383   36
629                     0.148   21
559                     0.300   35
684                     0.640   69

[614 rows x 8 columns]
```

[ ]:
```
x_test
```

[ ]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
661            1    199.0           76.0           43.0    155.0  42.9
122            2    107.0           74.0           30.0    100.0  33.6
113            4     76.0           62.0           29.0    155.0  34.0
14             5    166.0           72.0           19.0    175.0  25.8
529            0    111.0           65.0           29.0    155.0  24.6
..           …      …             …             …      …    …
476            2    105.0           80.0           45.0    191.0  33.7
482            4     85.0           58.0           22.0     49.0  27.8
230            4    142.0           86.0           29.0    155.0  44.0
527            3    116.0           74.0           15.0    105.0  26.3
380            1    107.0           72.0           30.0     82.0  30.8

     DiabetesPedigreeFunction  Age
661                     1.394   22
122                     0.404   23
113                     0.391   25
14                      0.587   51
529                     0.660   31
..                        …   …
476                     0.711   29
```

```
482                         0.306   28
230                         0.645   22
527                         0.107   24
380                         0.821   24

[154 rows x 8 columns]
```

Applying KNN

[ ]:
```
#feature Scaling

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[ ]:
```
#Loading KNN model

classifier = KNeighborsClassifier(n_neighbors = 11, p = 2, metric = 'euclidean')
```

[ ]:
```
#fitting the model

classifier.fit(x_train, y_train)
```

[ ]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

[ ]:
```
#making predictions

y_pred = classifier.predict(x_test)
y_pred
```

[ ]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[ ]:
```
#Evaluating the model

conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)
print(f1_score(y_test, y_pred))
```

```
[[94 13]
 [15 32]]
0.6956521739130436
```

[ ]:

```python
#Display the accuracy

print(accuracy_score(y_test, y_pred))
```

```
0.8181818181818182
```

# lab3-k-means

June 14, 2023

Customer Segmentation using K-means Algo

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

[16]:
```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/
 ↪Mail_Customers')
```

[17]:
```
data
```

[17]:
```
     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
..          …       …   …                 …                     …
195         196  Female   35                 120                      79
196         197  Female   45                 126                      28
197         198    Male   32                 126                      74
198         199    Male   32                 137                      18
199         200    Male   30                 137                      83

[200 rows x 5 columns]
```

[6]:
```
data.head()
```

[6]:
```
   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
```

[7]:
```
data.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
```

[8]:
```
data.shape
```

[8]:
```
(200, 5)
```

[9]:
```
data.isnull().sum()
```

[9]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

[10]:
```
data.describe()
```

[10]:

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| count | 200.000000 | 200.000000 | 200.000000 | 200.000000 |
| mean | 100.500000 | 38.850000 | 60.560000 | 50.200000 |
| std | 57.879185 | 13.969007 | 26.264721 | 25.823522 |
| min | 1.000000 | 18.000000 | 15.000000 | 1.000000 |
| 25% | 50.750000 | 28.750000 | 41.500000 | 34.750000 |
| 50% | 100.500000 | 36.000000 | 61.500000 | 50.000000 |
| 75% | 150.250000 | 49.000000 | 78.000000 | 73.000000 |
| max | 200.000000 | 70.000000 | 137.000000 | 99.000000 |

[18]:
```
data.Gender.value_counts()
```

[18]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

[19]:
```
#Customer_id column has no relevance therefore deleting it would be better
data = data.drop('CustomerID', axis = 1)
data.head()
```

[19]:

| | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| 0 | Male | 19 | 15 | 39 |
| 1 | Male | 21 | 15 | 81 |
| 2 | Female | 20 | 16 | 6 |
| 3 | Female | 23 | 16 | 77 |
| 4 | Female | 31 | 17 | 40 |

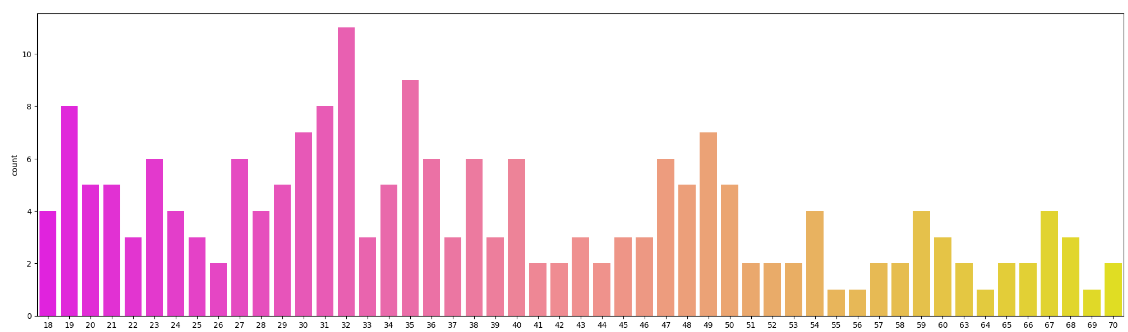
[20]:
```
#renaming the columns
data = data.rename(columns = {'Annual Income (k$)' : 'Annual_Income', 'Spending␣
↪Score (1-100)' : 'Spending_Score'})
data.head()
```

[20]:

| | Gender | Age | Annual_Income | Spending_Score |
|---|---|---|---|---|
| 0 | Male | 19 | 15 | 39 |
| 1 | Male | 21 | 15 | 81 |
| 2 | Female | 20 | 16 | 6 |
| 3 | Female | 23 | 16 | 77 |
| 4 | Female | 31 | 17 | 40 |

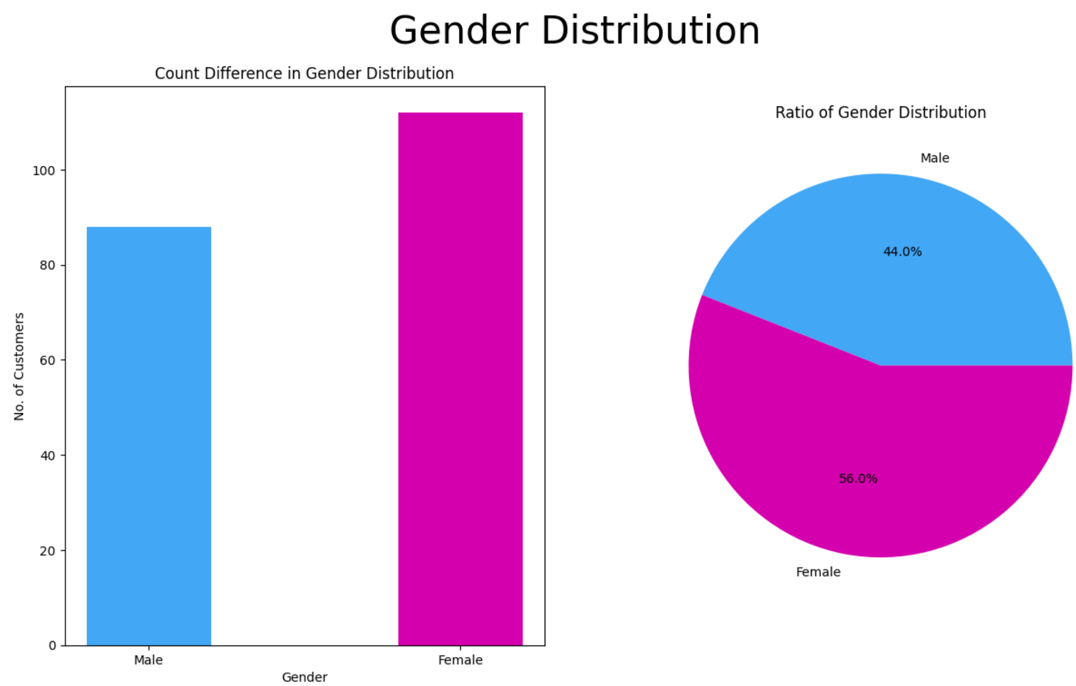
[22]:
```
plt.figure(figsize = (25,7))
sns.countplot(x = data['Age'], palette = 'spring')
```

[22]: <Axes: xlabel='Age', ylabel='count'>




Subplot

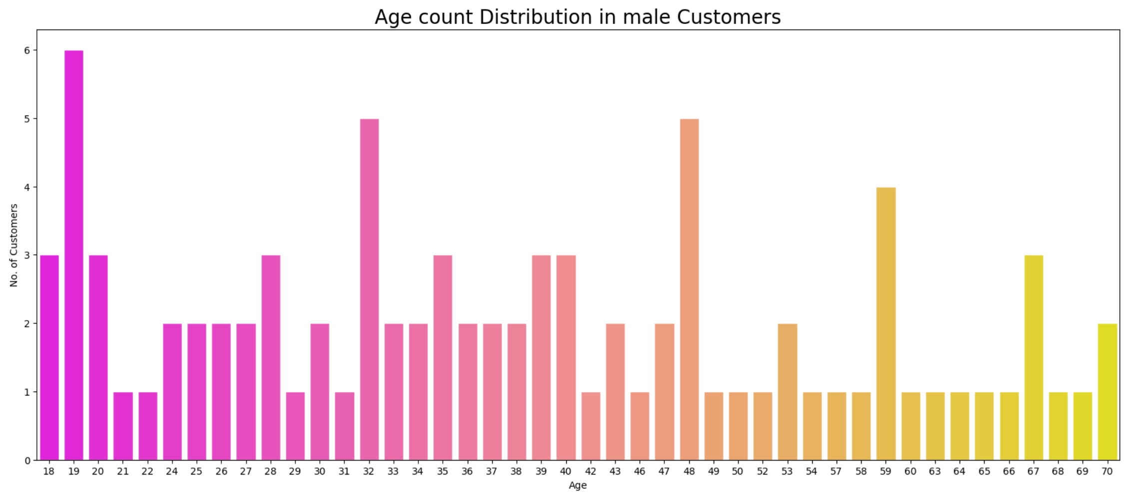
[26]:
```
#subplot
labels = data['Gender'].unique()
values = data['Gender'].value_counts(ascending = True)
fig, (ax0, ax1) = plt.subplots(ncols = 2, figsize = (15,8))

bar = ax0.bar(x=labels, height = values, width = 0.4, align = 'center', color =␣
↪['#42a7f5', '#d400ad'])
ax0.set(title = 'Count Difference in Gender Distribution', xlabel = 'Gender',␣
↪ylabel = 'No. of Customers')

ax1.pie(values, labels = labels, colors = ['#42a7f5', '#d400ad'], autopct = '%1.
↪1f%%')

ax1.set(title = 'Ratio of Gender Distribution')
fig.suptitle('Gender Distribution', fontsize = 30);
plt.show()
```



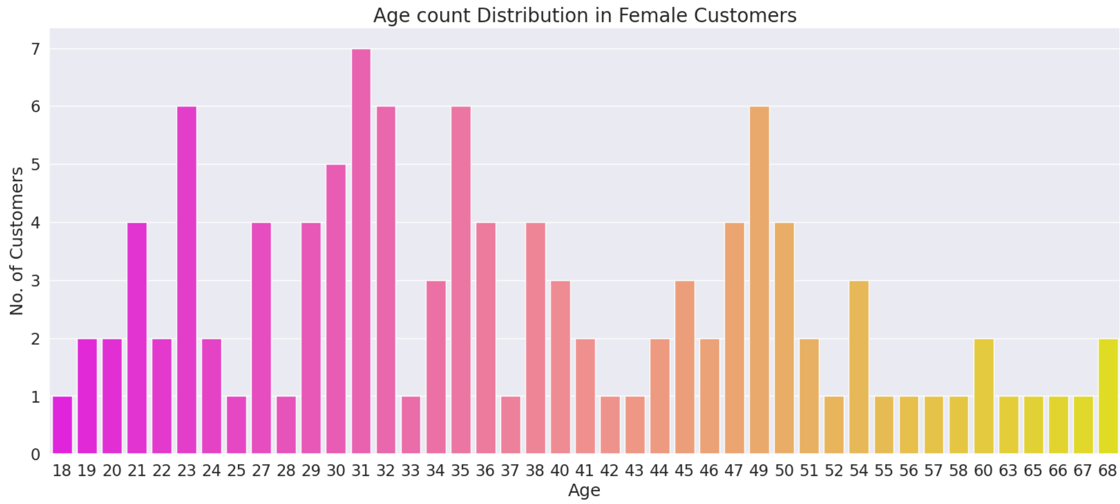

Countplot

[28]:
```
#Visualizing distribution of age count in male customers using a countplot
maxi = data[data['Gender'] == 'Male'].Age.value_counts().max()
mean = data[data['Gender'] == 'Male'].Age.value_counts().mean()
mini = data[data['Gender'] == 'Male'].Age.value_counts().min()

fig, ax = plt.subplots(figsize = (20, 8))
```

```
sns.set(font_scale = 1.5)
ax = sns.countplot(x = data[data['Gender'] == 'Male'].Age, palette = 'spring')
ax.set_ylabel('No. of Customers')
plt.title('Age count Distribution in male Customers', fontsize = 20)
plt.show()
```

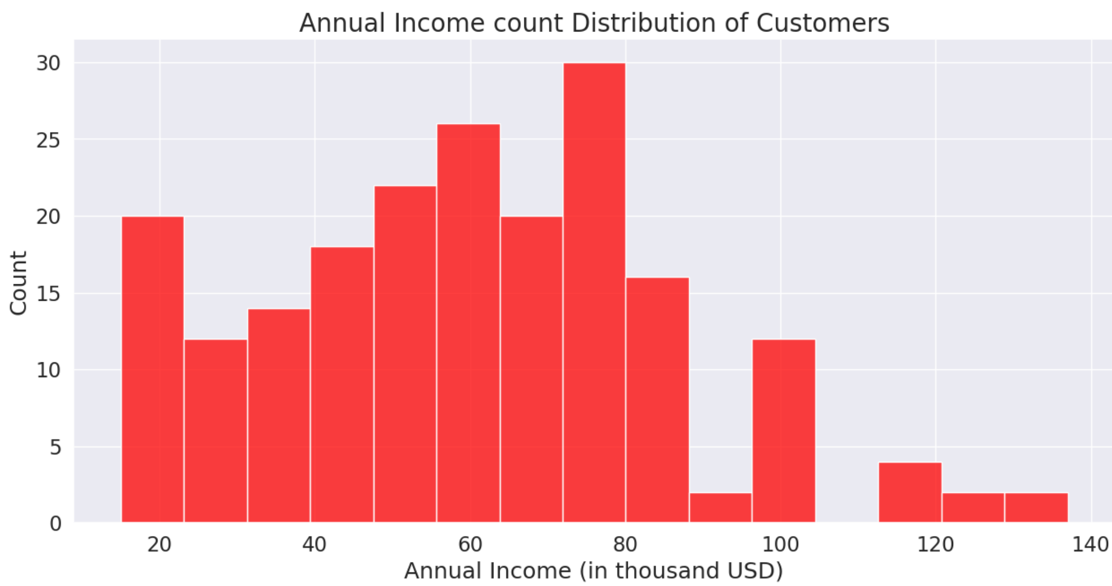



```
[29]: #Visualizing distribution of age count in Female customers using a countplot
maxi = data[data['Gender'] == 'Female'].Age.value_counts().max()
mean = data[data['Gender'] == 'Female'].Age.value_counts().mean()
mini = data[data['Gender'] == 'Female'].Age.value_counts().min()

fig, ax = plt.subplots(figsize = (20, 8))
sns.set(font_scale = 1.5)
ax = sns.countplot(x = data[data['Gender'] == 'Female'].Age, palette = 'spring')
ax.set_ylabel('No. of Customers')
plt.title('Age count Distribution in Female Customers', fontsize = 20)
plt.show()
```



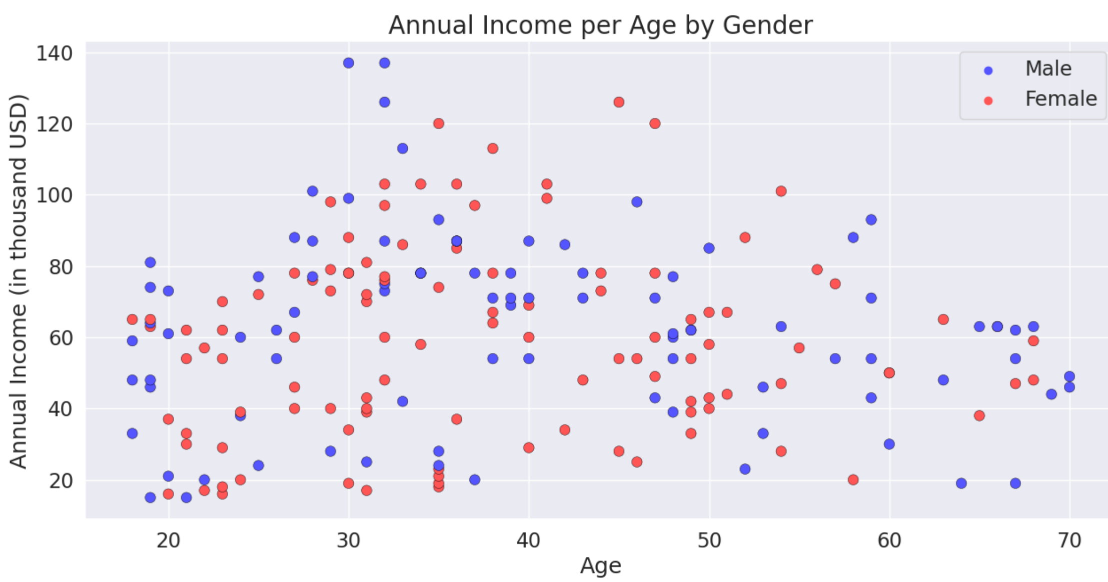

Histogram

[33]:
```
#Visualizing Annual Income count value distribution on a histogram
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale = 1.5)
ax = sns.histplot(data['Annual_Income'], bins = 15, ax = ax, color = 'red')
ax.set_xlabel('Annual Income (in thousand USD)')
plt.title('Annual Income count Distribution of Customers', fontsize = 20)
plt.show()
```




Scatter plot

```
[35]: #Visualizing annual Income per age by gender on a scatter plot.
fig, ax = plt.subplots(figsize = (15, 7))
sns.set(font_scale = 1.5)
ax = sns.scatterplot(y = data['Annual_Income'], x = data['Age'], hue =␣
 ↪data['Gender'], palette = 'seismic', s = 70, edgecolor = 'black', linewidth␣
 ↪= 0.3)
ax.set_ylabel('Annual Income (in thousand USD)')
ax.legend(loc = 'upper right')

plt.title('Annual Income per Age by Gender', fontsize = 20)
plt.show()
```




Applying K-Means

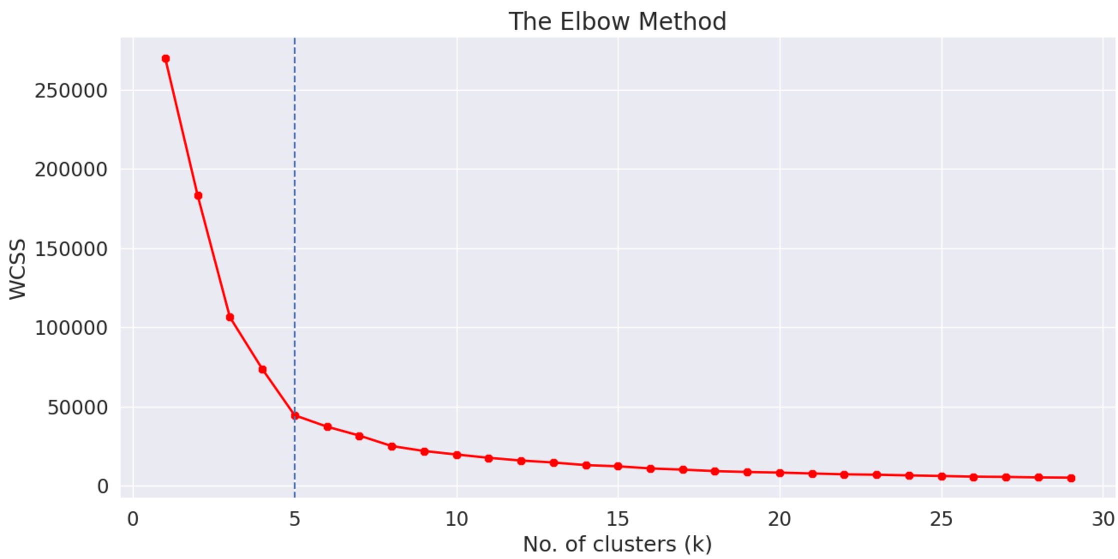
```
[37]: clustering_data = data.iloc[:, [2,3]]
```

```
[39]: #Determining No. of clusters Required within the cluster sum of square(wcss)
wcss=[]
for i in range(1, 30):
  km = KMeans(i, n_init = 10)
  km.fit(clustering_data)
  wcss.append(km.inertia_)
np.array(wcss)
```

```
[39]: array([269981.28      , 183499.07470289, 106348.37306211,  73679.78903949,
        44448.45544793,  37233.81451071,  31605.86838023,  25004.83031471,
        21850.16528259,  19657.7836087 ,  17549.69929191,  15838.71778551,
        14589.49268259,  12944.68210123,  12195.26383911,  10848.92599369,
```

```
       10091.99861776,   9145.20308513,   8599.15734127,   8217.44749164,
        7655.06695739,   7138.54206349,   6881.37190587,   6453.62853813,
        6074.25267313,   5644.77546349,   5489.3046398 ,   5184.55048285,
        5029.62883662])
```

[40]:
```
#Elbow Method
fig, ax = plt.subplots(figsize = (15, 7))
ax = plt.plot(range(1, 30), wcss, linewidth = 2, color = "red", marker = "8")
plt.axvline(x = 5, ls = '--')
plt.ylabel('WCSS')
plt.xlabel('No. of clusters (k)')
plt.title('The Elbow Method', fontsize = 20)
plt.show()
```




[41]:
```
#fitting
kms = KMeans(n_clusters = 5, init = 'k-means++')
kms.fit(clustering_data)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[41]:
```
KMeans(n_clusters=5)
```

[42]:
```
clusters = clustering_data.copy()
clusters['Cluster_Prediction'] = kms.fit_predict(clustering_data)
clusters.head()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[42]:

| | Annual_Income | Spending_Score | Cluster_Prediction |
|---|---|---|---|
| 0 | 15 | 39 | 0 |
| 1 | 15 | 81 | 2 |
| 2 | 16 | 6 | 0 |
| 3 | 16 | 77 | 2 |
| 4 | 17 | 40 | 0 |

# lab4-decisiontree

June 14, 2023

Decision Tree Classification oh Diabetes Dataset

[18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
```

[32]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
```

[ ]:
```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/diabetes_.
 ↪csv')
```

[33]:
```
data
```

[33]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …            …            …        …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
```

```
..                       …   …       …
763                      0.171   63         0
764                      0.340   27         0
765                      0.245   30         0
766                      0.349   47         1
767                      0.315   23         0

[768 rows x 9 columns]
```

[6]:
```
data.shape
```

[6]:
```
(768, 9)
```

[7]:
```
data.describe()
```

[7]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[34]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
```

```
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[35]:
```
data.isnull().sum()
```

[35]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[36]:
```
#feature variable

x = data.drop(['Outcome'], axis = 1)
x
```

[36]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …             …             …       …    …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        …    …
763                     0.171   63
764                     0.340   27
```

```
765                    0.245   30
766                    0.349   47
767                    0.315   23

[768 rows x 8 columns]
```

[37]:

```
#target variable

y = data.Outcome
y
```

[37]:

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

[38]:

```
plt.figure(figsize = (14, 7))
sns.heatmap(data.corr(), annot = True, cmap = 'Purples')
plt.show()
```

Splitting data

[39]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,␣
 ↪random_state = 1)
```

[40]:
```
#create Decision Tree Classifier object
model = DecisionTreeClassifier()

#Train Decision Tree Classifier
model = model.fit(x_train, y_train)

#predict the response for the test dataset
y_pred = model.predict(x_test)
```

[41]:
```
#evaluation using Accuracy score
from sklearn import metrics

#import scikit-learn matrics module for accuracy calculation
print("Accuracy: ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy:  67.53246753246754
```

[42]:
```
#Evaluation using confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

[42]:
```
array([[77, 22],
       [28, 27]])
```

[44]:
```
#Evaluation using classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.78      0.75        99
           1       0.55      0.49      0.52        55

    accuracy                           0.68       154
   macro avg       0.64      0.63      0.64       154
weighted avg       0.67      0.68      0.67       154
```

[45]:
```
#import modules for visualizing Decision trees
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
```

[46]:
```
features = x.columns
features
```

[46]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'],
      dtype='object')
```

[48]:
```
dot_data = StringIO()
export_graphviz(model, out_file = dot_data, filled = True, rounded = True,␣
 ↪special_characters = True, feature_names = features, class_names =['0','1'])

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes_set.png')
Image(graph.create_png())
```

Output hidden; open in https://colab.research.google.com to view.

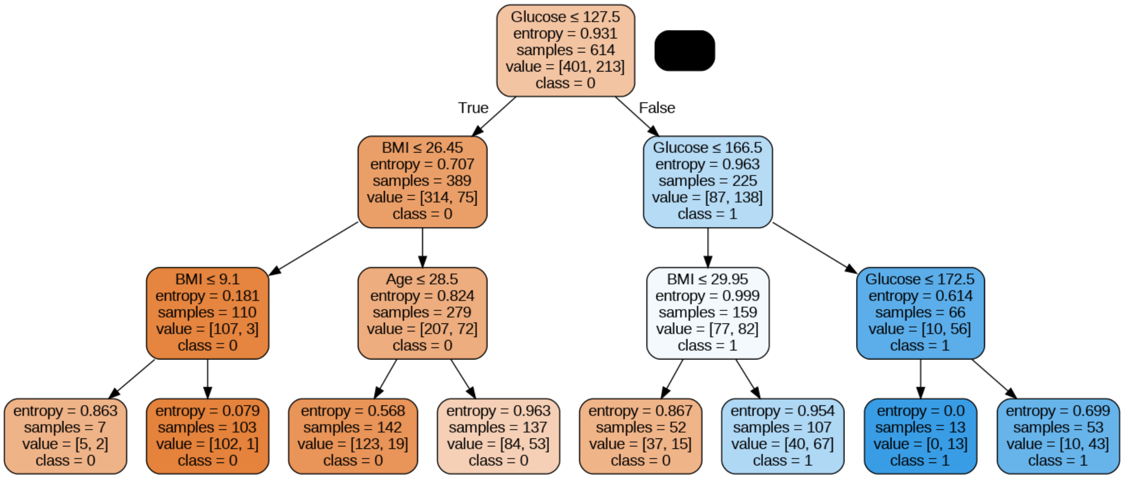
[49]:
```
#Create Decision Tree Classifier object
model = DecisionTreeClassifier(criterion = "entropy", max_depth = 3)

#Train Decision Tree Classifier
model = model.fit(x_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)

#model Accuracy
print("Accuracy:" , metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 79.87012987012987
```

The classification rate increased to 79.87%. which is better accuracy than the preious model

[51]:
```
#Better Decision Treee visualization
dot_data = StringIO()
export_graphviz(model, out_file = dot_data, filled = True, rounded = True,␣
 ↪special_characters = True, feature_names = features, class_names =['0','1'])

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes_set.png')
Image(graph.create_png())
```

[51]:


Glucose ≤ 127.5
entropy = 0.931
samples = 614
value = [401, 213]
class = 0
True
False
BMI ≤ 26.45
entropy = 0.707
samples = 389
value = [314, 75]
class = 0
Glucose ≤ 166.5
entropy = 0.963
samples = 225
value = [87, 138]
class = 1
BMI ≤ 9.1
entropy = 0.181
samples = 110
value = [107, 3]
class = 0
Age ≤ 28.5
entropy = 0.824
samples = 279
value = [207, 72]
class = 0
BMI ≤ 29.95
entropy = 0.999
samples = 159
value = [77, 82]
class = 1
Glucose ≤ 172.5
entropy = 0.614
samples = 66
value = [10, 56]
class = 1
entropy = 0.863
samples = 7
value = [5, 2]
class = 0
entropy = 0.079
samples = 103
value = [102, 1]
class = 0
entropy = 0.568
samples = 142
value = [123, 19]
class = 0
entropy = 0.963
samples = 137
value = [84, 53]
class = 0
entropy = 0.867
samples = 52
value = [37, 15]
class = 0
entropy = 0.954
samples = 107
value = [40, 67]
class = 1
entropy = 0.0
samples = 13
value = [0, 13]
class = 1
entropy = 0.699
samples = 53
value = [10, 43]
class = 1

```
[ ]:
```

# lab5-svm

June 14, 2023

SVM

[54]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn import metrics
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

[ ]:
```
test_tmp = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/
  ↪SalaryData_Test.csv')
train_tmp = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/
  ↪SalaryData_Train.csv')
```

[ ]:
```
df_tmp = test_tmp.append(train_tmp)
```

```
<ipython-input-4-6842bd4f2a35>:1: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  df_tmp = test_tmp.append(train_tmp)
```

[ ]:
```
test = test_tmp.copy()
train = train_tmp.copy()
```

[ ]:
```
test
```

[ ]:
```
       age    workclass     education  educationno        maritalstatus  \
0       25      Private          11th            7        Never-married
1       38      Private       HS-grad            9   Married-civ-spouse
2       28    Local-gov    Assoc-acdm           12   Married-civ-spouse
3       44      Private  Some-college           10   Married-civ-spouse
4       34      Private          10th            6        Never-married
```

```
…     …          …          …      …                  …
15055  33      Private   Bachelors            13       Never-married
15056  39      Private   Bachelors            13            Divorced
15057  38      Private   Bachelors            13  Married-civ-spouse
15058  44      Private   Bachelors            13            Divorced
15059  35  Self-emp-inc   Bachelors           13  Married-civ-spouse

              occupation    relationship                  race      sex  \
0      Machine-op-inspct       Own-child                 Black     Male
1        Farming-fishing         Husband                 White     Male
2        Protective-serv         Husband                 White     Male
3      Machine-op-inspct         Husband                 Black     Male
4          Other-service   Not-in-family                 White     Male
…                    …               …                  …       …
15055     Prof-specialty       Own-child                 White     Male
15056     Prof-specialty   Not-in-family                 White   Female
15057     Prof-specialty         Husband                 White     Male
15058       Adm-clerical       Own-child    Asian-Pac-Islander     Male
15059    Exec-managerial         Husband                 White     Male

       capitalgain  capitalloss  hoursperweek         native  Salary
0                0            0            40  United-States   <=50K
1                0            0            50  United-States   <=50K
2                0            0            40  United-States    >50K
3             7688            0            40  United-States    >50K
4                0            0            30  United-States   <=50K
…           …            …             …              …       …
15055            0            0            40  United-States   <=50K
15056            0            0            36  United-States   <=50K
15057            0            0            50  United-States   <=50K
15058         5455            0            40  United-States   <=50K
15059            0            0            60  United-States    >50K

[15060 rows x 14 columns]
```

[ ]:
```
train
```

[ ]:
```
       age          workclass   education  educationno        maritalstatus  \
0       39          State-gov   Bachelors           13        Never-married
1       50   Self-emp-not-inc   Bachelors           13   Married-civ-spouse
2       38            Private     HS-grad            9             Divorced
3       53            Private        11th            7   Married-civ-spouse
4       28            Private   Bachelors           13   Married-civ-spouse
…     …                  …          …          …                   …
30156   27            Private  Assoc-acdm           12   Married-civ-spouse
30157   40            Private     HS-grad            9   Married-civ-spouse
30158   58            Private     HS-grad            9              Widowed
```

```
30159   22            Private      HS-grad            9        Never-married
30160   52       Self-emp-inc      HS-grad            9   Married-civ-spouse

               occupation    relationship    race      sex  capitalgain  \
0            Adm-clerical   Not-in-family   White     Male         2174
1         Exec-managerial         Husband   White     Male            0
2       Handlers-cleaners   Not-in-family   White     Male            0
3       Handlers-cleaners         Husband   Black     Male            0
4          Prof-specialty            Wife   Black   Female            0
…                     …               …     …       …           …
30156        Tech-support            Wife   White   Female            0
30157   Machine-op-inspct         Husband   White     Male            0
30158        Adm-clerical       Unmarried   White   Female            0
30159        Adm-clerical       Own-child   White     Male            0
30160     Exec-managerial            Wife   White   Female        15024

       capitalloss  hoursperweek         native  Salary
0                0            40  United-States   <=50K
1                0            13  United-States   <=50K
2                0            40  United-States   <=50K
3                0            40  United-States   <=50K
4                0            40           Cuba   <=50K
…              …             …             …      …
30156            0            38  United-States   <=50K
30157            0            40  United-States    >50K
30158            0            40  United-States   <=50K
30159            0            20  United-States   <=50K
30160            0            40  United-States    >50K

[30161 rows x 14 columns]
```

[ ]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15060 entries, 0 to 15059
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            15060 non-null  int64
 1   workclass      15060 non-null  object
 2   education      15060 non-null  object
 3   educationno    15060 non-null  int64
 4   maritalstatus  15060 non-null  object
 5   occupation     15060 non-null  object
 6   relationship   15060 non-null  object
 7   race           15060 non-null  object
 8   sex            15060 non-null  object
```

```
 9   capitalgain    15060 non-null  int64
 10  capitalloss    15060 non-null  int64
 11  hoursperweek   15060 non-null  int64
 12  native         15060 non-null  object
 13  Salary         15060 non-null  object
dtypes: int64(5), object(9)
memory usage: 1.6+ MB
```

[ ]:
```
test.describe()
```

[ ]:
```
                age  educationno   capitalgain   capitalloss  hoursperweek
count  15060.000000  15060.000000  15060.000000  15060.000000  15060.000000
mean      38.768327     10.112749   1120.301594     89.041899     40.951594
std       13.380676      2.558727   7703.181842    406.283245     12.062831
min       17.000000      1.000000      0.000000      0.000000      1.000000
25%       28.000000      9.000000      0.000000      0.000000     40.000000
50%       37.000000     10.000000      0.000000      0.000000     40.000000
75%       48.000000     13.000000      0.000000      0.000000     45.000000
max       90.000000     16.000000  99999.000000   3770.000000     99.000000
```

Label encoding(To convert string into int)

[ ]:
```
str_c = ["workclass", "education", "maritalstatus", "occupation",␣
 ↪"relationship", "race", "sex", "native"]
```

[ ]:
```
#Label encoder can be used to normalize labels. used to transform non-numerical␣
 ↪labels to numerical labels.
number = LabelEncoder()
```

[ ]:
```
for i in str_c:
  train[i] = number.fit_transform(train[i])
  test[i] = number.fit_transform(test[i])
```

[ ]:
```
test.head()
```

[ ]:
```
   age  workclass  education  educationno  maritalstatus  occupation  \
0   25          2          1            7              4           6
1   38          2         11            9              2           4
2   28          1          7           12              2          10
3   44          2         15           10              2           6
4   34          2          0            6              4           7

   relationship  race  sex  capitalgain  capitalloss  hoursperweek  native  \
0             3     2    1            0            0            40      37
1             0     4    1            0            0            50      37
2             0     4    1            0            0            40      37
3             0     2    1         7688            0            40      37
```

```
4              1     4    1            0            0            30       37

   Salary
0   <=50K
1   <=50K
2    >50K
3    >50K
4   <=50K
```

[ ]:
```python
train.head()
```

[ ]:
```
   age  workclass  education  educationno  maritalstatus  occupation  \
0   39          5          9           13              4           0
1   50          4          9           13              2           3
2   38          2         11            9              0           5
3   53          2          1            7              2           5
4   28          2          9           13              2           9

   relationship  race  sex  capitalgain  capitalloss  hoursperweek  native  \
0             1     4    1         2174            0            40      37
1             0     4    1            0            0            13      37
2             1     4    1            0            0            40      37
3             0     2    1            0            0            40      37
4             5     2    0            0            0            40       4

   Salary
0   <=50K
1   <=50K
2   <=50K
3   <=50K
4   <=50K
```

Mapping

[ ]:
```python
mapping = {' >50K': 1, ' <=50K': 2}
```

[ ]:
```python
train = train.replace({'Salary': mapping})
test = test.replace({'Salary': mapping})
```

[ ]:
```python
df = train.append(test)
```

```
<ipython-input-26-c75fa8e72363>:1: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  df = train.append(test)
```

[ ]:
```python
df
```

[ ]:
```
          age  workclass  education  educationno  maritalstatus  occupation  \
0          39          5          9           13              4           0
1          50          4          9           13              2           3
2          38          2         11            9              0           5
3          53          2          1            7              2           5
4          28          2          9           13              2           9
…     …         …         …          …            …          …
15055      33          2          9           13              4           9
15056      39          2          9           13              0           9
15057      38          2          9           13              2           9
15058      44          2          9           13              0           0
15059      35          3          9           13              2           3

       relationship  race  sex  capitalgain  capitalloss  hoursperweek  \
0                 1     4    1         2174            0            40
1                 0     4    1            0            0            13
2                 1     4    1            0            0            40
3                 0     2    1            0            0            40
4                 5     2    0            0            0            40
…              …     …   …          …          …           …
15055             3     4    1            0            0            40
15056             1     4    0            0            0            36
15057             0     4    1            0            0            50
15058             3     1    1         5455            0            40
15059             0     4    1            0            0            60

       native  Salary
0          37       2
1          37       2
2          37       2
3          37       2
4           4       2
…       …      …
15055      37       2
15056      37       2
15057      37       2
15058      37       2
15059      37       1

[45221 rows x 14 columns]
```

[ ]:
```
df1 = df.copy()
```

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

[ ]:
```
   age  workclass  education  educationno  maritalstatus  occupation  \
0   39          5          9           13              4           0
```

```
1   50          4          9           13              2            3
2   38          2         11            9              0            5
3   53          2          1            7              2            5
4   28          2          9           13              2            9

   relationship  race  sex  capitalgain  capitalloss  hoursperweek  native  \
0             1     4    1         2174            0            40      37
1             0     4    1            0            0            13      37
2             1     4    1            0            0            40      37
3             0     2    1            0            0            40      37
4             5     2    0            0            0            40       4

   Salary
0       2
1       2
2       2
3       2
4       2
```

[ ]:
```
df1.shape
```

[ ]:
```
(45221, 14)
```

[ ]:
```
df1.describe(include='all')
```

[ ]:
```
                age     workclass     education   educationno  maritalstatus  \
count  45221.000000  45221.000000  45221.000000  45221.000000   45221.000000
mean      38.548086      2.204507     10.313217     10.118463       2.585148
std       13.217981      0.958132      3.816992      2.552909       1.500460
min       17.000000      0.000000      0.000000      1.000000       0.000000
25%       28.000000      2.000000      9.000000      9.000000       2.000000
50%       37.000000      2.000000     11.000000     10.000000       2.000000
75%       47.000000      2.000000     12.000000     13.000000       4.000000
max       90.000000      6.000000     15.000000     16.000000       6.000000

         occupation  relationship          race           sex   capitalgain  \
count  45221.000000  45221.000000  45221.000000  45221.000000  45221.000000
mean       5.969572      1.412684      3.680281      0.675062   1101.454700
std        4.026444      1.597242      0.832361      0.468357   7506.511295
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        2.000000      0.000000      4.000000      0.000000      0.000000
50%        6.000000      1.000000      4.000000      1.000000      0.000000
75%        9.000000      3.000000      4.000000      1.000000      0.000000
max       13.000000      5.000000      4.000000      1.000000  99999.000000

        capitalloss  hoursperweek        native        Salary
count  45221.000000  45221.000000  45221.000000  45221.000000
```

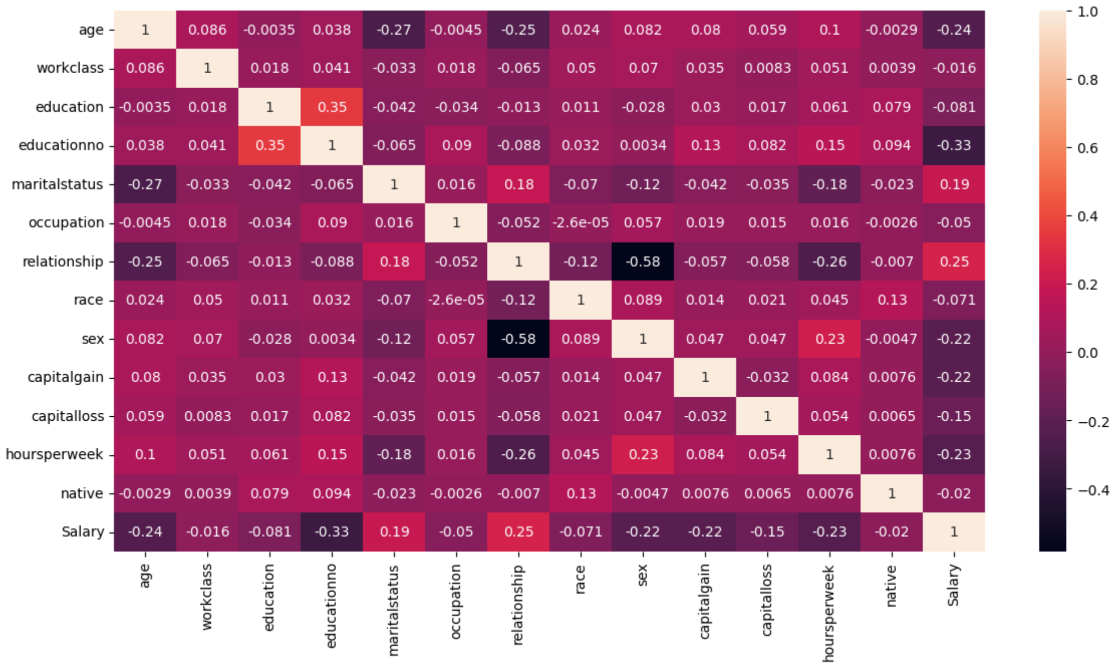
| | | | | |
|---|---|---|---|---|
| mean | 88.548617 | 40.938038 | 35.431503 | 1.752151 |
| std | 404.838249 | 12.007640 | 5.931380 | 0.431769 |
| min | 0.000000 | 1.000000 | 0.000000 | 1.000000 |
| 25% | 0.000000 | 40.000000 | 37.000000 | 2.000000 |
| 50% | 0.000000 | 40.000000 | 37.000000 | 2.000000 |
| 75% | 0.000000 | 45.000000 | 37.000000 | 2.000000 |
| max | 4356.000000 | 99.000000 | 39.000000 | 2.000000 |

[ ]:
```
df1.isnull().sum()
```

[ ]:
```
age              0
workclass        0
education        0
educationno      0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
capitalgain      0
capitalloss      0
hoursperweek     0
native           0
Salary           0
dtype: int64
```

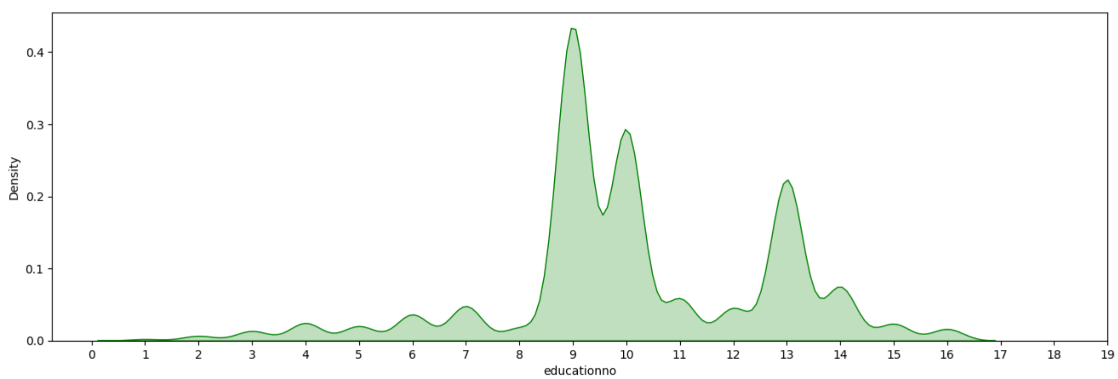
[37]:
```
plt.figure(figsize = (14, 7))
sns.heatmap(df1.corr(), annot = True)
```

[37]: <Axes: >



```
[39]: plt.figure(figsize = (16, 5))
print("Skew: {}".format(df1['educationno'].skew()))
print("kurtosis: {}".format(df1['educationno'].kurtosis()))
ax = sns.kdeplot(df1['educationno'], fill = True, color = 'g')
plt.xticks([i for i in range(0, 20, 1)])
plt.show()
```

```
Skew: -0.31062061074424
kurtosis: 0.6350448194491634
```

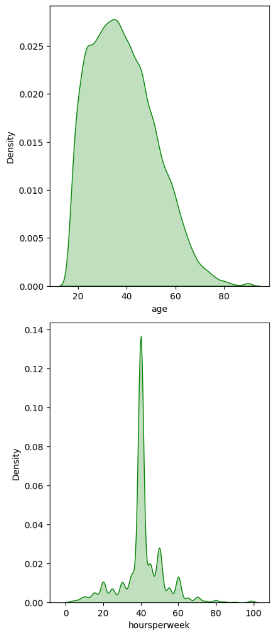


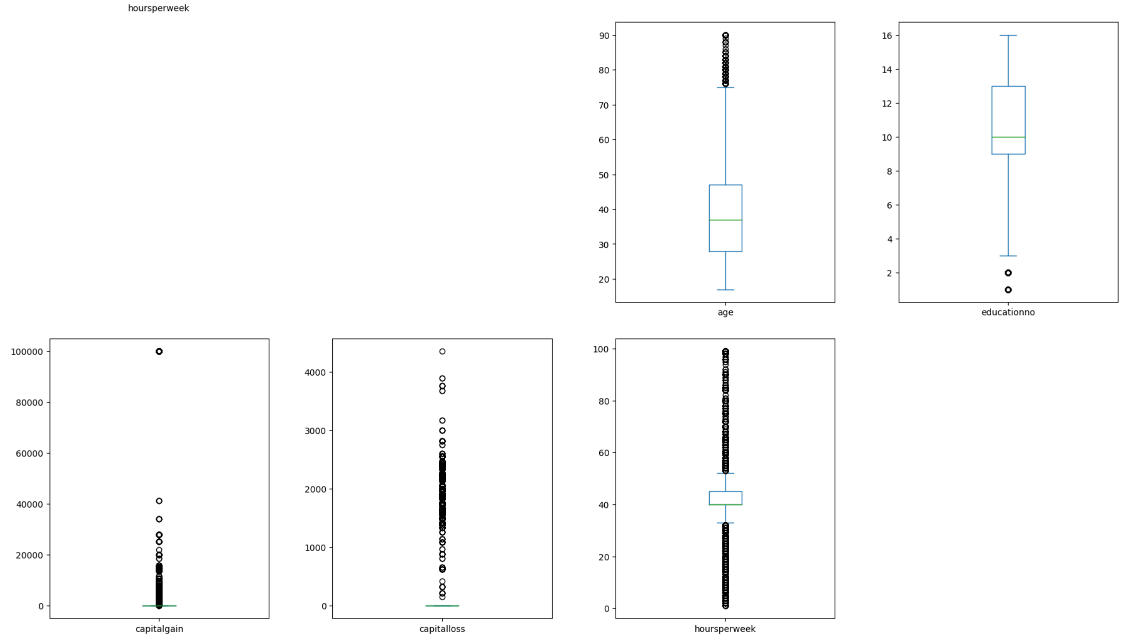

The data is negatively skewed and has low kurtosis. most of the people have education no of years between 9 - 10

[42]:

```python
dfa = df_tmp[df_tmp.columns[0:13]]
num_columns = dfa.select_dtypes(exclude = 'object').columns.tolist()
plt.figure(figsize = (18,40))
for i, col in enumerate(num_columns, 1):
  plt.subplot(8, 4, i)
  sns.kdeplot(df[col], color = 'g', shade = True)
  plt.subplot(8, 4, i+10)
  df[col].plot.box()
plt.tight_layout()
plt.show()
num_data = df[num_columns]
pd.DataFrame(data = [num_data.skew(), num_data.kurtosis()], index =␣
 ↪['skewness', 'kurtosis'])
```

```
<ipython-input-42-2c0b37f22eae>:6: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], color = 'g', shade = True)
<ipython-input-42-2c0b37f22eae>:6: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], color = 'g', shade = True)
<ipython-input-42-2c0b37f22eae>:6: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], color = 'g', shade = True)
<ipython-input-42-2c0b37f22eae>:6: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], color = 'g', shade = True)
<ipython-input-42-2c0b37f22eae>:6: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], color = 'g', shade = True)
```




[42]:

| | age | educationno | capitalgain | capitalloss | hoursperweek |
|---|---|---|---|---|---|
| skewness | 0.532784 | -0.310621 | 11.788871 | 4.517536 | 0.340536 |
| kurtosis | -0.155931 | 0.635045 | 150.147899 | 19.376085 | 3.201287 |

SVM. Normalization

[43]:
```
#svm
col = df1.columns
```

[44]:
```
x_train = train[col[0:13]]
y_train = train[col[13]]
x_test = test[col[0:13]]
y_test = test[col[13]]
```

[46]:
```
def norm_func(i):
  x = (i-i.min())/(i.max()-i.min())
  return(x)
```

[47]:
```
x_train = norm_func(x_train)
x_test = norm_func(x_test)
```

[55]:
```
#using linear svm
model_linear = SVC(kernel = "linear")
model_linear.fit(x_train, y_train)
pred_test_linear = model_linear.predict(x_test)
print("Accuracy : ", metrics.accuracy_score(y_test, pred_test_linear))
```

```
Accuracy :  0.8097609561752988
```

[56]:
```
#using Poly svm
model_poly = SVC(kernel = "poly")
model_poly.fit(x_train, y_train)
pred_test_poly = model_poly.predict(x_test)
print("Accuracy : ", metrics.accuracy_score(y_test, pred_test_poly))
```

```
Accuracy :  0.8435590969455511
```

[58]:
```
#using RBF svm
model_rbf = SVC(kernel = "rbf")
model_rbf.fit(x_train, y_train)
pred_test_rbf = model_rbf.predict(x_test)
print("Accuracy : ", metrics.accuracy_score(y_test, pred_test_rbf))
```

```
Accuracy :  0.8432934926958832
```

[59]:
```
#using sigmoid svm
model_sigmoid = SVC(kernel = "sigmoid")
model_sigmoid.fit(x_train, y_train)
pred_test_sigmoid = model_sigmoid.predict(x_test)
print("Accuracy : ", metrics.accuracy_score(y_test, pred_test_sigmoid))
```

```
Accuracy :  0.5768924302788845
```

#conclusion : Poly model gives the best accuracy

# lab6-linearregression

June 14, 2023

Linear Regression

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

[ ]:
```
customers = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/
 ↪Ecommerce Customers')
```

[ ]:
```
customers
```

[ ]:
```
                             Email  \
0        mstephenson@fernandez.com
1                hduke@hotmail.com
2                 pallen@yahoo.com
3          riverarebecca@gmail.com
4    mstephens@davidson-herman.com
..                             …
495   lewisjessica@craig-evans.com
496            katrina56@gmail.com
497             dale88@hotmail.com
498            cwilson@hotmail.com
499      hannahwilson@davidson.com

                                                Address            Avatar  \
0        835 Frank Tunnel\nWrightmouth, MI 82180-9605            Violet
1      4547 Archer Common\nDiazchester, CA 06566-8576         DarkGreen
2    24645 Valerie Unions Suite 582\nCobbborough, D…            Bisque
3     1414 David Throughway\nPort Jason, OH 22070-1220       SaddleBrown
4    14023 Rodriguez Passage\nPort Jacobville, PR 3…  MediumAquaMarine
..                                                …                 …
495  4483 Jones Motorway Suite 872\nLake Jamiefurt,…               Tan
```

```
496  172 Owen Divide Suite 497\nWest Richard, CA 19320     PaleVioletRed
497  0787 Andrews Ranch Apt. 633\nSouth Chadburgh, …          Cornsilk
498  680 Jennifer Lodge Apt. 808\nBrendachester, TX…              Teal
499  49791 Rachel Heights Apt. 898\nEast Drewboroug…       DarkMagenta

     Avg. Session Length  Time on App  Time on Website  Length of Membership  \
0              34.497268    12.655651        39.577668              4.082621
1              31.926272    11.109461        37.268959              2.664034
2              33.000915    11.330278        37.110597              4.104543
3              34.305557    13.717514        36.721283              3.120179
4              33.330673    12.795189        37.536653              4.446308
..                   …          …              …                    …
495            33.237660    13.566160        36.417985              3.746573
496            34.702529    11.695736        37.190268              3.576526
497            32.646777    11.499409        38.332576              4.958264
498            33.322501    12.391423        36.840086              2.336485
499            33.715981    12.418808        35.771016              2.735160

     Yearly Amount Spent
0             587.951054
1             392.204933
2             487.547505
3             581.852344
4             599.406092
..                   …
495           573.847438
496           529.049004
497           551.620145
498           456.469510
499           497.778642

[500 rows x 8 columns]
```

[ ]:
```
customers.head()
```

[ ]:
```
                           Email  \
0      mstephenson@fernandez.com
1              hduke@hotmail.com
2               pallen@yahoo.com
3        riverarebecca@gmail.com
4  mstephens@davidson-herman.com

                                             Address            Avatar  \
0       835 Frank Tunnel\nWrightmouth, MI 82180-9605            Violet
1     4547 Archer Common\nDiazchester, CA 06566-8576         DarkGreen
2  24645 Valerie Unions Suite 582\nCobbborough, D…            Bisque
3   1414 David Throughway\nPort Jason, OH 22070-1220       SaddleBrown
```

```
4  14023 Rodriguez Passage\nPort Jacobville, PR 3…  MediumAquaMarine

   Avg. Session Length  Time on App  Time on Website  Length of Membership  \
0            34.497268    12.655651        39.577668              4.082621
1            31.926272    11.109461        37.268959              2.664034
2            33.000915    11.330278        37.110597              4.104543
3            34.305557    13.717514        36.721283              3.120179
4            33.330673    12.795189        37.536653              4.446308

   Yearly Amount Spent
0           587.951054
1           392.204933
2           487.547505
3           581.852344
4           599.406092
```

[ ]:
```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Email                 500 non-null    object
 1   Address               500 non-null    object
 2   Avatar                500 non-null    object
 3   Avg. Session Length   500 non-null    float64
 4   Time on App           500 non-null    float64
 5   Time on Website       500 non-null    float64
 6   Length of Membership  500 non-null    float64
 7   Yearly Amount Spent   500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

[ ]:
```
customers.describe()
```

[ ]:
```
       Avg. Session Length  Time on App  Time on Website  \
count           500.000000   500.000000       500.000000
mean             33.053194    12.052488        37.060445
std               0.992563     0.994216         1.010489
min              29.532429     8.508152        33.913847
25%              32.341822    11.388153        36.349257
50%              33.082008    11.983231        37.069367
75%              33.711985    12.753850        37.716432
max              36.139662    15.126994        40.005182

       Length of Membership  Yearly Amount Spent
```

```
count      500.000000  500.000000
mean         3.533462  499.314038
std          0.999278   79.314782
min          0.269901  256.670582
25%          2.930450  445.038277
50%          3.533975  498.887875
75%          4.126502  549.313828
max          6.922689  765.518462
```

[ ]:
```
customers.columns
```

[ ]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
      dtype='object')
```

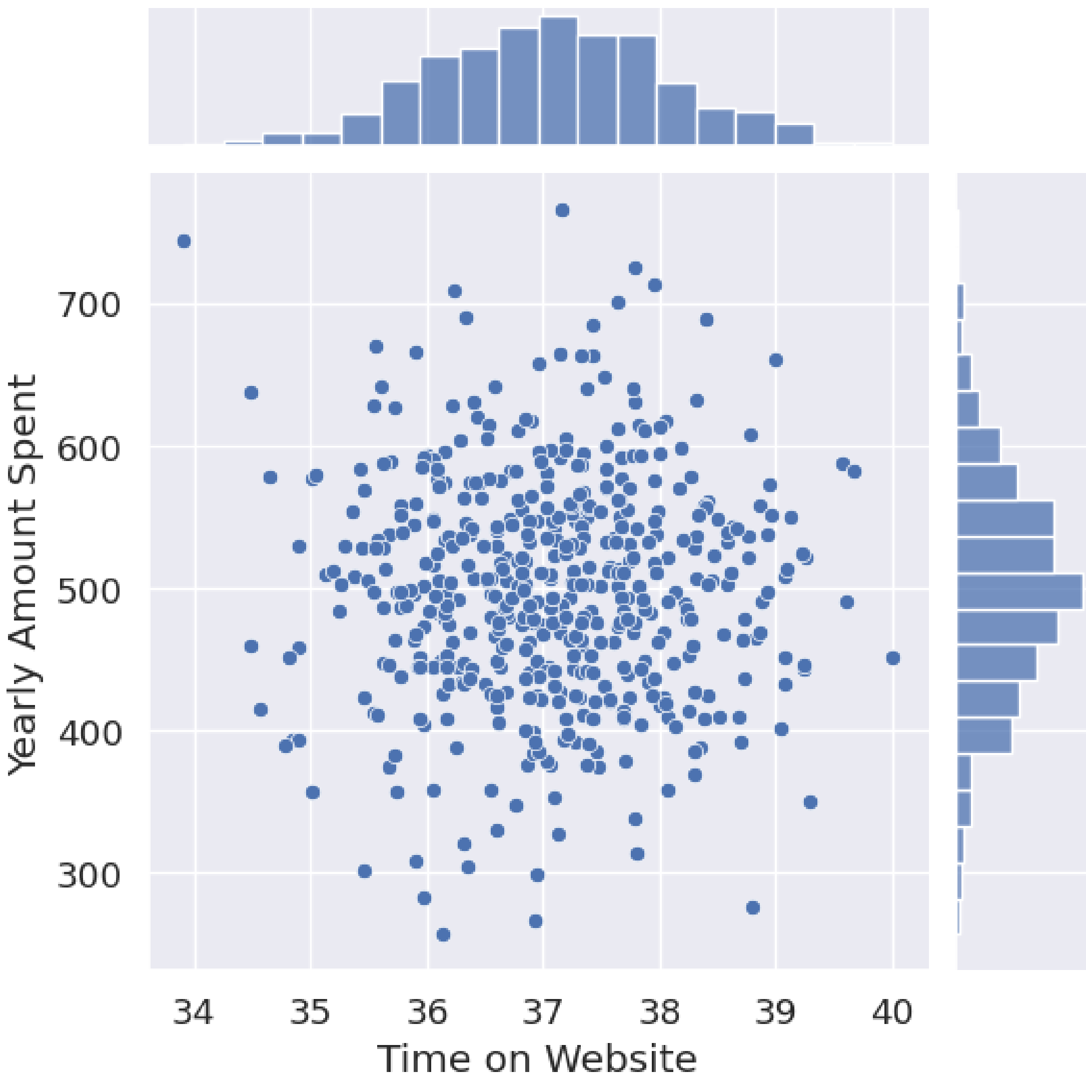
[ ]:
```
customers.shape
```

[ ]:
```
(500, 8)
```

[ ]:
```
plt.figure(figsize = (12,8))
sns.set(font_scale = 1.2)
sns.jointplot(data = customers, x = 'Time on Website', y = 'Yearly Amount␣
 ↪Spent')
```

[ ]:
```
<seaborn.axisgrid.JointGrid at 0x7f0b123af7f0>
```

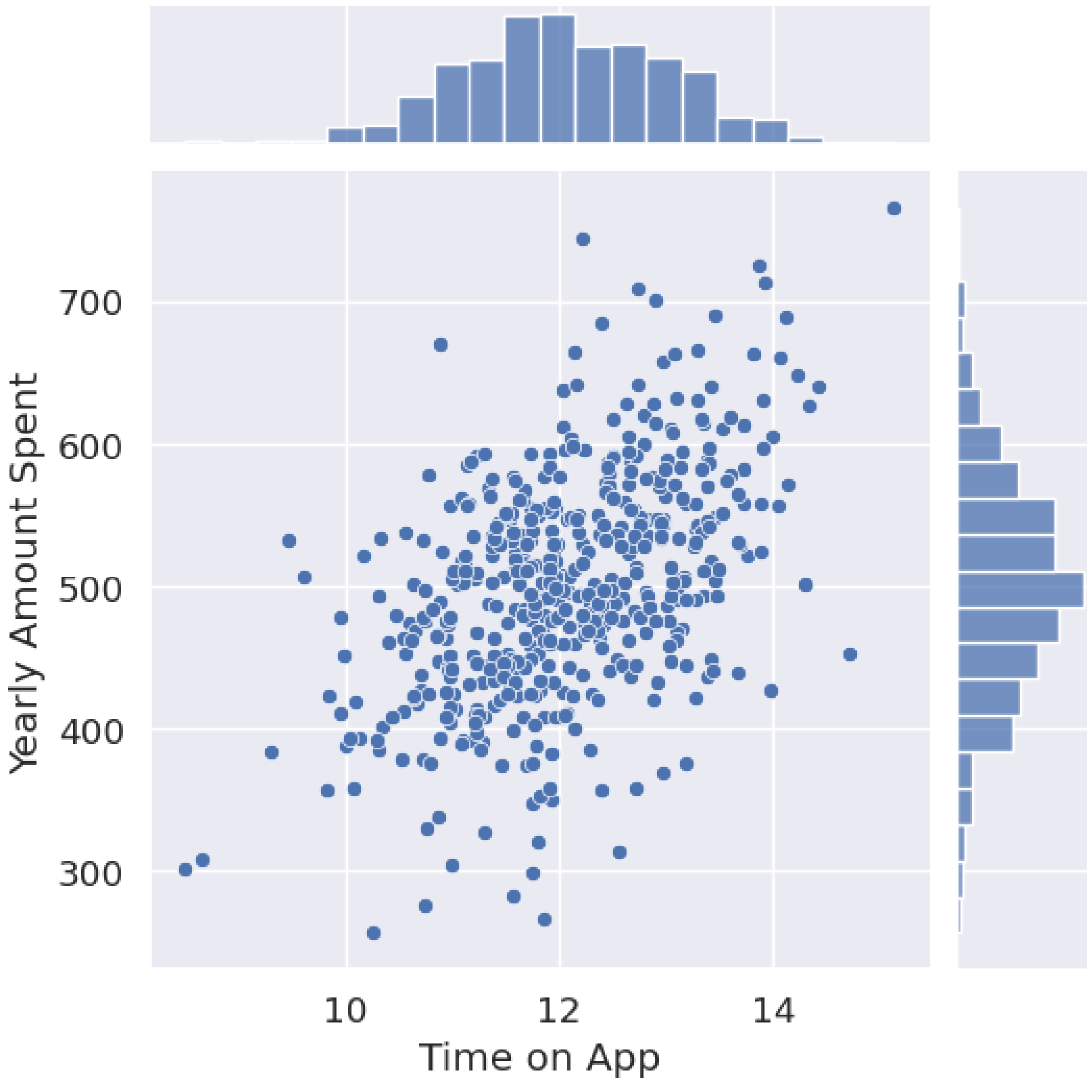
```
<Figure size 1200x800 with 0 Axes>
```




[ ]:
```
plt.figure(figsize = (12,8))
sns.set(font_scale = 1.2)
sns.jointplot(data = customers, x = 'Time on App', y = 'Yearly Amount Spent')
```

[ ]: <seaborn.axisgrid.JointGrid at 0x7f0b11f56ad0>

<Figure size 1200x800 with 0 Axes>




[ ]:
```
customers[['Time on Website', 'Yearly Amount Spent']].corr()
```

[ ]:

| | Time on Website | Yearly Amount Spent |
|---|---|---|
| Time on Website | 1.000000 | -0.002641 |
| Yearly Amount Spent | -0.002641 | 1.000000 |

[ ]:
```
customers[['Time on App', 'Yearly Amount Spent']].corr()
```

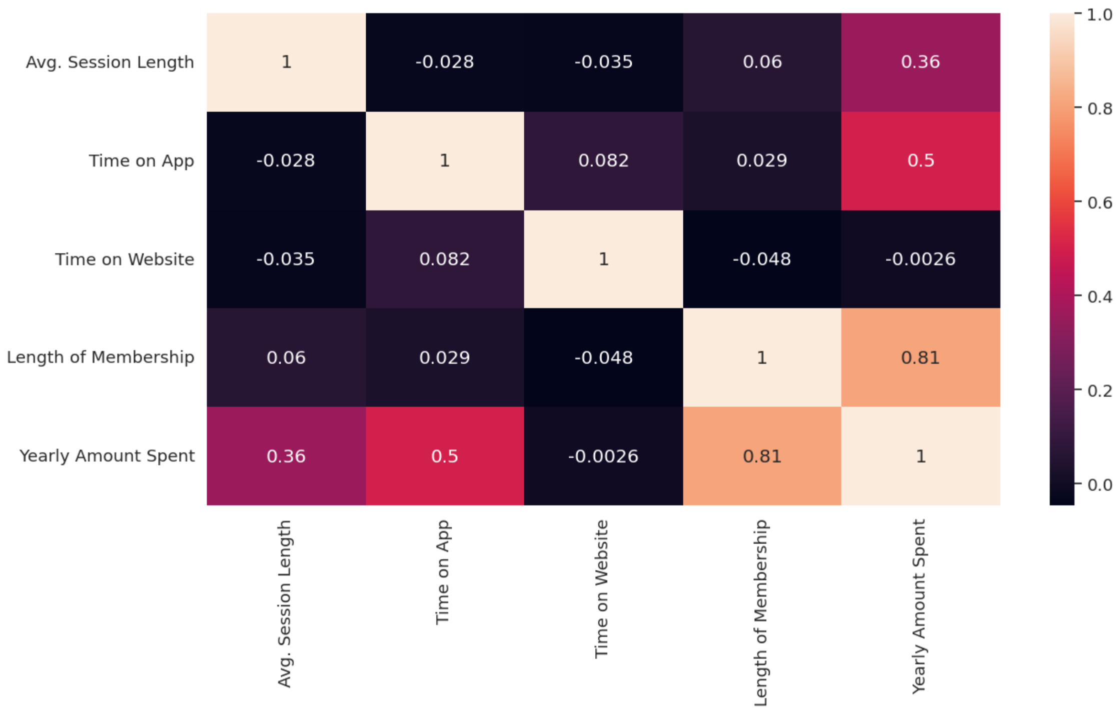
[ ]:

| | Time on App | Yearly Amount Spent |
|---|---|---|
| Time on App | 1.000000 | 0.499328 |
| Yearly Amount Spent | 0.499328 | 1.000000 |

[ ]:
```
plt.figure(figsize = (14, 7))
sns.heatmap(customers.corr(), annot = True)
```

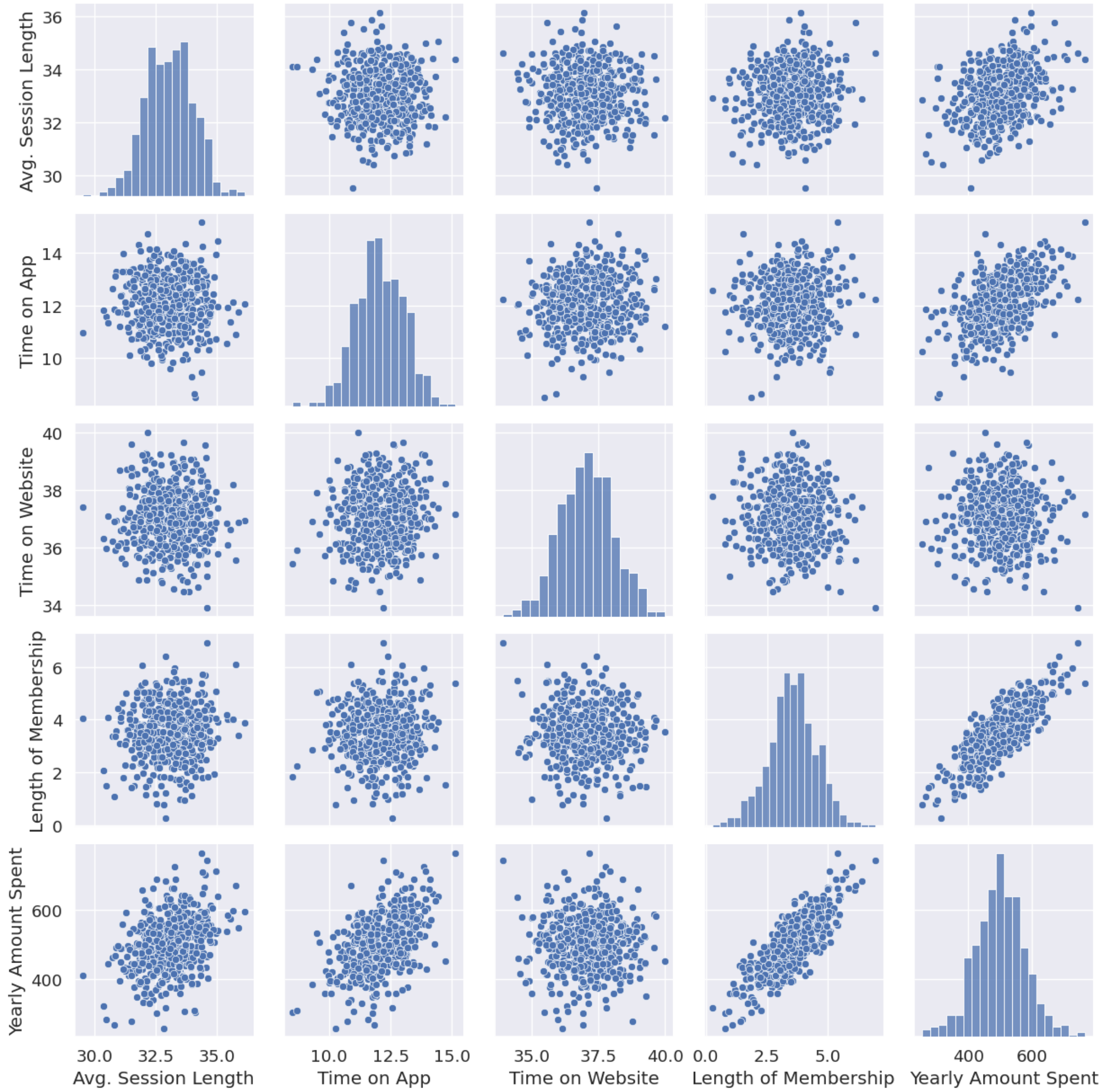
```
<ipython-input-22-a925dab6c6ac>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(customers.corr(), annot = True)
```

[ ]:
```
<Axes: >
```



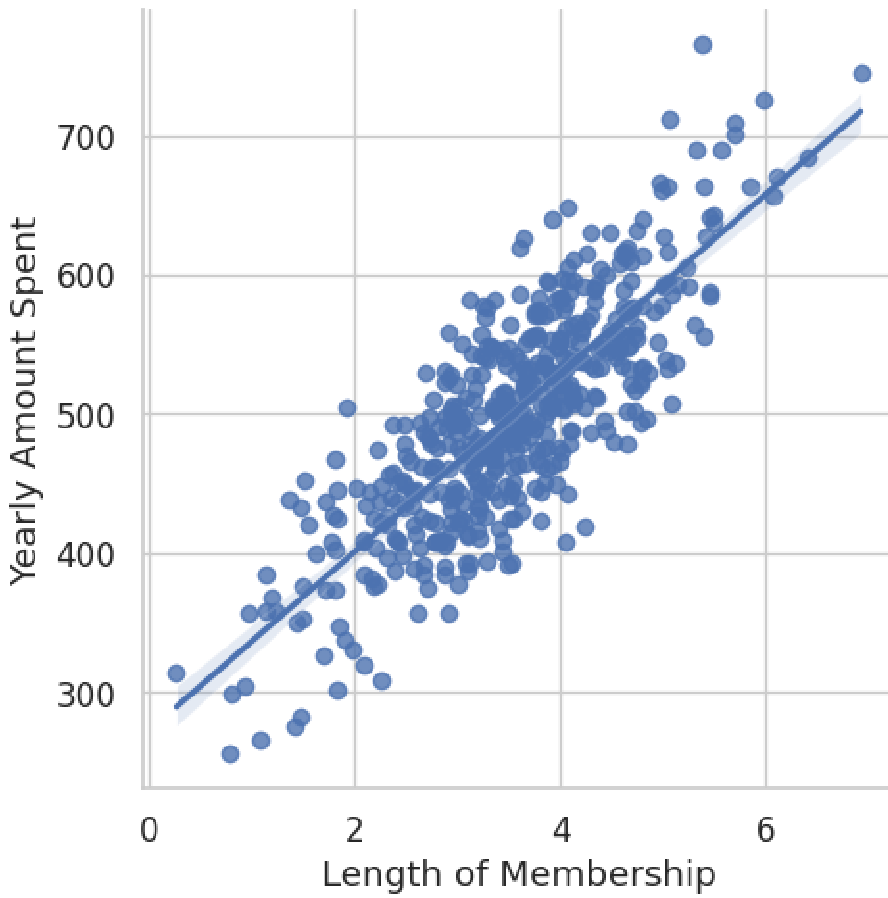

[ ]:
```
sns.pairplot(customers)
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f0b0e256b60>
```




[ ]:
```
sns.set(font_scale = 1.1)
sns.set_style('whitegrid')
sns.lmplot(y = "Yearly Amount Spent", x = "Length of Membership", data =␣
 ↪customers)
```

[ ]: <seaborn.axisgrid.FacetGrid at 0x7f0b0da7d4b0>




[ ]:
```
customers[['Length of Membership','Yearly Amount Spent']].corr()
```

[ ]:
```
                      Length of Membership  Yearly Amount Spent
Length of Membership              1.000000             0.809084
Yearly Amount Spent               0.809084             1.000000
```

Extracting and Splitting

[ ]:
```
x = customers[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length␣
↪of Membership']]
y = customers['Yearly Amount Spent']
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
↪random_state = 101)
```

[ ]:
```
x_train
```

```
[ ]:      Avg. Session Length  Time on App  Time on Website  Length of Membership
    202            31.525752    11.340036        37.039514              3.811248
    428            31.862741    14.039867        37.022269              3.738225
    392            33.258238    11.514949        37.128039              4.662845
    86             33.877779    12.517666        37.151921              2.669942
    443            33.025020    12.504220        37.645839              4.051382
    ..                   …            …              …                    …
    63             32.789773    11.670066        37.408748              3.414688
    326            33.217188    10.999684        38.442767              4.243813
    337            31.827979    12.461147        37.428997              2.974737
    11             33.879361    11.584783        37.087926              3.713209
    351            32.189845    11.386776        38.197483              4.808320

    [350 rows x 4 columns]
```

```
[ ]: x_test
```

```
[ ]:      Avg. Session Length  Time on App  Time on Website  Length of Membership
    18             32.187812    14.715388        38.244115              1.516576
    361            32.077590    10.347877        39.045156              3.434560
    104            31.389585    10.994224        38.074452              3.428860
    4              33.330673    12.795189        37.536653              4.446308
    156            32.294642    12.443048        37.327848              5.084861
    ..                   …            …              …                    …
    147            32.255901    10.480507        37.338670              4.514122
    346            32.765665    12.506548        35.823467              3.126509
    423            33.128693    10.398458        36.683393              3.859818
    17             32.338899    12.013195        38.385137              2.420806
    259            32.096109    10.804891        37.372762              2.699562

    [150 rows x 4 columns]
```

```
[ ]: y_train
```

```
[ ]: 202    443.965627
    428    556.298141
    392    549.131573
    86     487.379306
    443    561.516532
              …
    63     483.159721
    326    505.230068
    337    440.002748
    11     522.337405
    351    533.396554
    Name: Yearly Amount Spent, Length: 350, dtype: float64
```

[ ]:
```
y_test
```

[ ]:
```
18     452.315675
361    401.033135
104    410.069611
4      599.406092
156    586.155870
          …
147    479.731938
346    488.387526
423    461.112248
17     407.704548
259    375.398455
Name: Yearly Amount Spent, Length: 150, dtype: float64
```

[ ]:
```
#Training model
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

[ ]:
```
#Train and fit lm on the training data
lm.fit(x_train, y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
#Evaluating our model
predictions = lm.predict(x_test)
predictions
```

[ ]:
```
array([456.44186104, 402.72005312, 409.2531539 , 591.4310343 ,
       590.01437275, 548.82396607, 577.59737969, 715.44428115,
       473.7893446 , 545.9211364 , 337.8580314 , 500.38506697,
       552.93478041, 409.6038964 , 765.52590754, 545.83973731,
       693.25969124, 507.32416226, 573.10533175, 573.2076631 ,
       397.44989709, 555.0985107 , 458.19868141, 482.66899911,
       559.2655959 , 413.00946082, 532.25727408, 377.65464817,
       535.0209653 , 447.80070905, 595.54339577, 667.14347072,
       511.96042791, 573.30433971, 505.02260887, 565.30254655,
       460.38785393, 449.74727868, 422.87193429, 456.55615271,
       598.10493696, 449.64517443, 615.34948995, 511.88078685,
       504.37568058, 515.95249276, 568.64597718, 551.61444684,
       356.5552241 , 464.9759817 , 481.66007708, 534.2220025 ,
       256.28674001, 505.30810714, 520.01844434, 315.0298707 ,
       501.98080155, 387.03842642, 472.97419543, 432.8704675 ,
       539.79082198, 590.03070739, 752.86997652, 558.27858232,
       523.71988382, 431.77690078, 425.38411902, 518.75571466,
       641.9667215 , 481.84855126, 549.69830187, 380.93738919,
       555.18178277, 403.43054276, 472.52458887, 501.82927633,
```

```
       473.5561656 , 456.76720365, 554.74980563, 702.96835044,
       534.68884588, 619.18843136, 500.11974127, 559.43899225,
       574.8730604 , 505.09183544, 529.9537559 , 479.20749452,
       424.78407899, 452.20986599, 525.74178343, 556.60674724,
       425.7142882 , 588.8473985 , 490.77053065, 562.56866231,
       495.75782933, 445.17937217, 456.64011682, 537.98437395,
       367.06451757, 421.12767301, 551.59651363, 528.26019754,
       493.47639211, 495.28105313, 519.81827269, 461.15666582,
       528.8711677 , 442.89818166, 543.20201646, 350.07871481,
       401.49148567, 606.87291134, 577.04816561, 524.50431281,
       554.11225704, 507.93347015, 505.35674292, 371.65146821,
       342.37232987, 634.43998975, 523.46931378, 532.7831345 ,
       574.59948331, 435.57455636, 599.92586678, 487.24017405,
       457.66383406, 425.25959495, 331.81731213, 443.70458331,
       563.47279005, 466.14764208, 463.51837671, 381.29445432,
       411.88795623, 473.48087683, 573.31745784, 417.55430913,
       543.50149858, 547.81091537, 547.62977348, 450.99057409,
       561.50896321, 478.30076589, 484.41029555, 457.59099941,
       411.52657592, 375.47900638])
```

[ ]:
```
from sklearn import metrics
metrics.mean_squared_error(y_test, predictions)
```

[ ]:
```
79.81305165097427
```

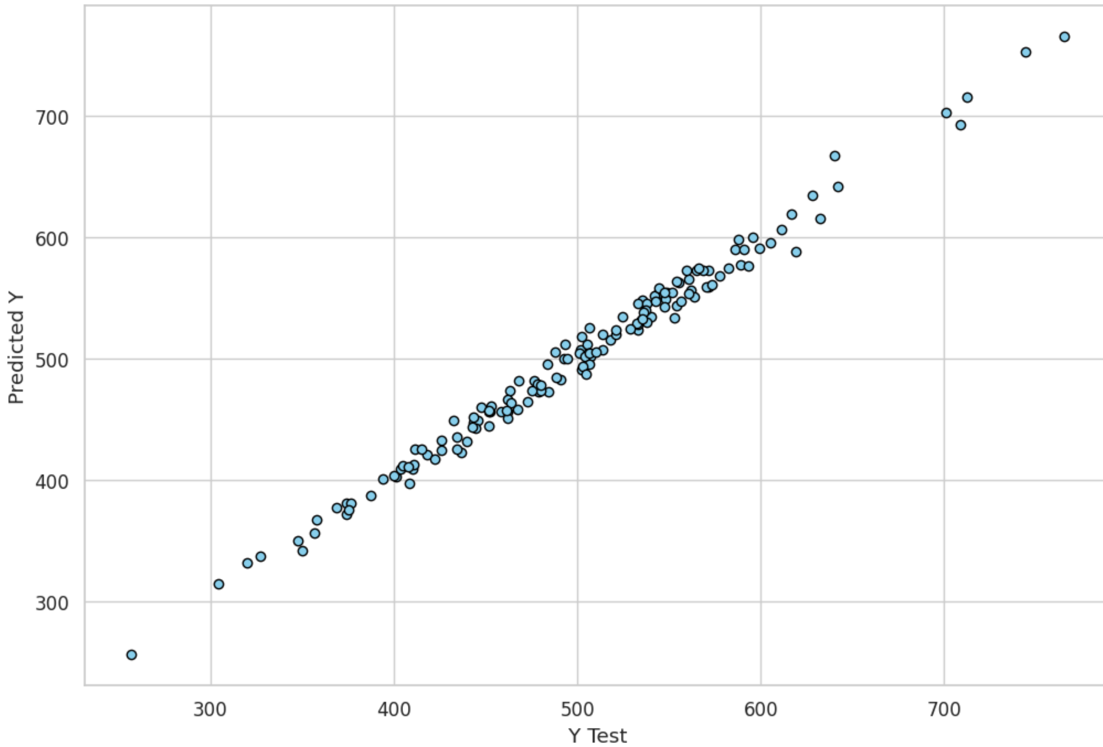
[ ]:
```
metrics.explained_variance_score(y_test, predictions)
```

[ ]:
```
0.9890771231889606
```

the model explains approximately 99% of the variance in the data

[40]:
```
#create a scatterplot of the real test values versus the predicted values.
plt.figure(figsize = (12, 8))
plt.scatter(x = y_test, y = predictions, color = 'skyblue', edgecolors =␣
 ↪'black')
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

[40]:
```
Text(0, 0.5, 'Predicted Y')
```




[42]:
```
#print out the coefficient of model
coef_df = pd.DataFrame(lm.coef_, x.columns, columns = ['coefficient'])
coef_df
```

[42]:

| | coefficient |
|---|---|
| Avg. Session Length | 25.981550 |
| Time on App | 38.590159 |
| Time on Website | 0.190405 |
| Length of Membership | 61.279097 |

#interpretations holding all other feature fixed, a 1 unit increase in session length is assosiated with an increase of $25.98 spent per year.

holding all other feature fixed, a 1 unit increase in Time on App is assosiated with an increase of $38.59 spent per year.

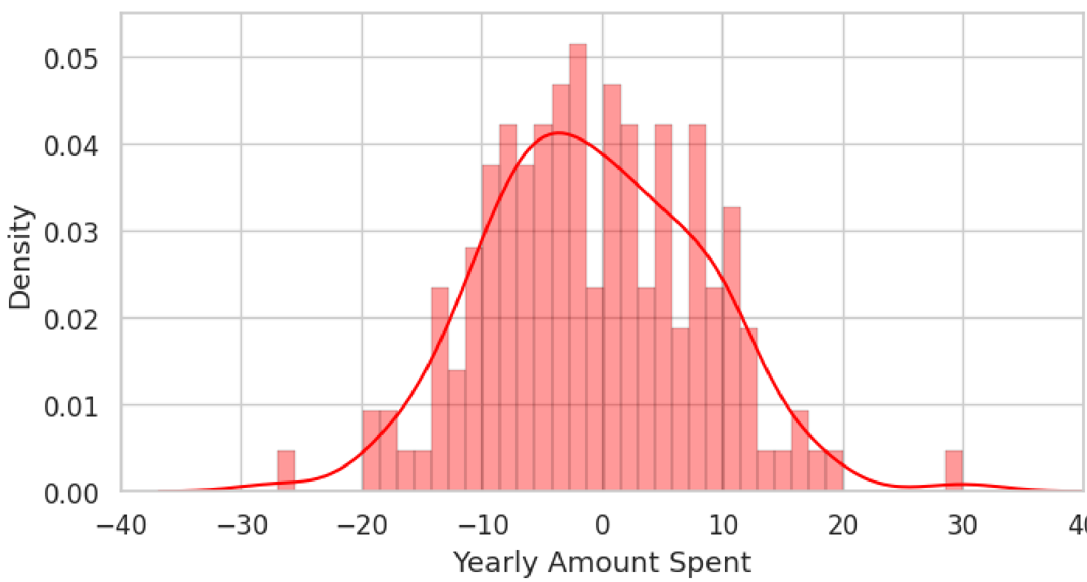
[43]:
```
rscore = lm.score(x, y)
rscore
```

[43]: 0.9842727142336021

[44]:
```
plt.figure(figsize = (8, 4))
ax = sns.distplot((y_test-predictions), bins = 40, color = 'red', hist_kws =␣
↪dict(edgecolor = 'black', linewidth = 0.3))
```

```
ax.set(xlim = (-40, 40))
ax.set(ylim = (0, 0.055));
```

```
<ipython-input-44-58e68c37ccaa>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot((y_test-predictions), bins = 40, color = 'red', hist_kws =
dict(edgecolor = 'black', linewidth = 0.3))
```




We can see that the yearly income increases with a greater capacity if customers spend time on App. The company should thus focus more on their mobile app, since the app performs better for increases in the yearly sales Amount.

[ ]:

```

```

# drug-decisiontree

June 14, 2023

[4]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
```

[7]:
```python
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

[8]:
```python
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/drug200.csv')
```

[9]:
```python
data
```

[9]:
```
     Age Sex      BP Cholesterol  Na_to_K   Drug
0     23   F    HIGH        HIGH   25.355  drugY
1     47   M     LOW        HIGH   13.093  drugC
2     47   M     LOW        HIGH   10.114  drugC
3     28   F  NORMAL        HIGH    7.798  drugX
4     61   F     LOW        HIGH   18.043  drugY
..   …  ..     …         …       …     …
195   56   F     LOW        HIGH   11.567  drugC
196   16   M     LOW        HIGH   12.006  drugC
197   52   M  NORMAL        HIGH    9.894  drugX
198   23   M  NORMAL      NORMAL   14.020  drugX
199   40   F     LOW      NORMAL   11.349  drugX

[200 rows x 6 columns]
```

[ ]:
```python
data.shape
```

[ ]:
```
(200, 6)
```

[ ]:
```python
data.describe()
```

[ ]:
```
              Age     Na_to_K
count  200.000000  200.000000
mean    44.315000   16.084485
std     16.544315    7.223956
min     15.000000    6.269000
25%     31.000000   10.445500
50%     45.000000   13.936500
75%     58.000000   19.380000
max     74.000000   38.247000
```

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

[ ]:
```
   Age Sex      BP Cholesterol  Na_to_K   Drug
0   23   F    HIGH        HIGH   25.355  drugY
1   47   M     LOW        HIGH   13.093  drugC
2   47   M     LOW        HIGH   10.114  drugC
3   28   F  NORMAL        HIGH    7.798  drugX
4   61   F     LOW        HIGH   18.043  drugY
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null    object
 3   Cholesterol  200 non-null    object
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

[10]:
```
data.isnull().sum()
```

[10]:
```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

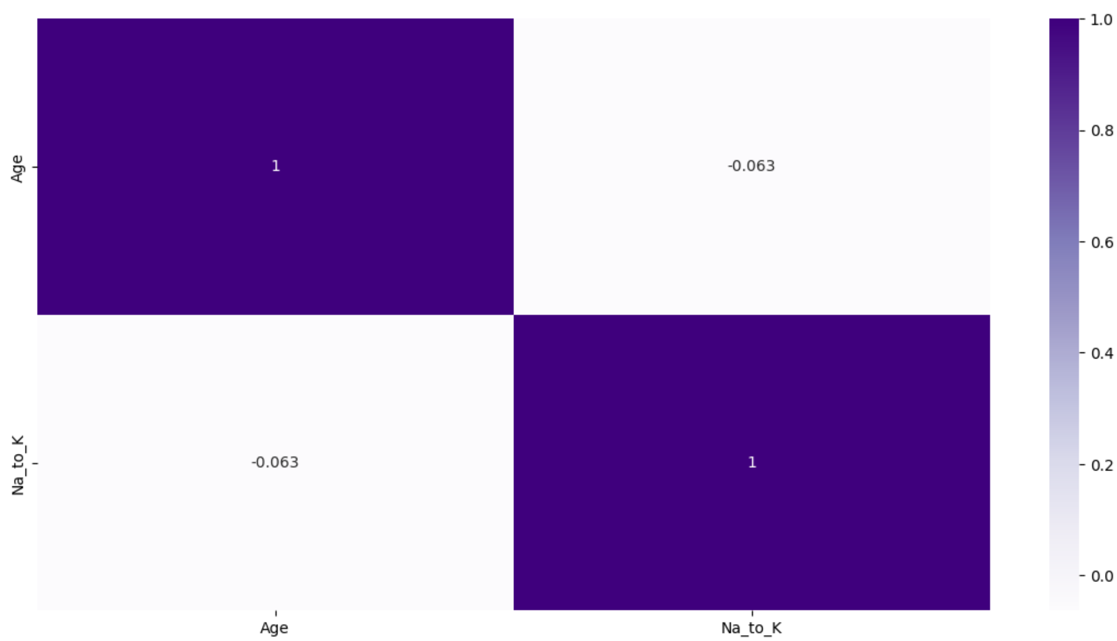
[24]:
```
#Converting datatype
x = data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
```

```
from sklearn import preprocessing
sex = preprocessing.LabelEncoder()
sex.fit(['F','M'])
x[:,1] = sex.transform(x[:,1])

BP = preprocessing.LabelEncoder()
BP.fit(['LOW', 'NORMAL', 'HIGH'])
x[:,2] = BP.transform(x[:,2])

chol = preprocessing.LabelEncoder()
chol.fit(['NORMAL', 'HIGH'])
x[:,3] = chol.transform(x[:,3])
```

[12]:
```
y = data.Drug
```

[14]:
```
plt.figure(figsize = (14, 7))
sns.heatmap(data.corr(), annot = True, cmap = 'Purples')
plt.show()
```

```
<ipython-input-14-389b52427130>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(data.corr(), annot = True, cmap = 'Purples')
```



[15]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,␣
 ↪random_state = 1)
```

[16]:
```
#create Decision Tree Classifier object
model = DecisionTreeClassifier()

#Train Decision Tree Classifier
model = model.fit(x_train, y_train)

#predict the response for the test dataset
y_pred = model.predict(x_test)
```

[17]:
```
#evaluation using Accuracy score
from sklearn import metrics

#import scikit-learn matrics module for accuracy calculation
print("Accuracy: ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy:  100.0
```

[18]:
```
#Evaluation using confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

[18]:
```
array([[ 4,  0,  0,  0,  0],
       [ 0,  2,  0,  0,  0],
       [ 0,  0,  4,  0,  0],
       [ 0,  0,  0, 13,  0],
       [ 0,  0,  0,  0, 17]])
```

[19]:
```
#Evaluation using classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       drugA       1.00      1.00      1.00         4
       drugB       1.00      1.00      1.00         2
       drugC       1.00      1.00      1.00         4
       drugX       1.00      1.00      1.00        13
       drugY       1.00      1.00      1.00        17

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

[20]:

```

```

[22]:

```

```