Dheepthi Priyangha CB.EN.U4CSE20217

PART-C

Q4

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_r
from sklearn.ensemble import RandomForestClassifier
from datascience import *
from sklearn.model_selection import cross_val_score
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
%matplotlib inline
```

Data Validation

```
data=pd.read_csv("/content/ckd_full.csv")
data.head()
```

	Age	Blood Pressure	Specific Gravity	Albumin	Sugar	Red Blood Cells	Pus Cell	Pus Cell clumps	Bacteria	Blood Glucose Random	...
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...

5 rows × 25 columns

```
#Data Cleaning

data['Hypertension'] = data['Hypertension'].replace(to_replace={'yes':1,'no':0
data['Diabetes Mellitus'] = data['Diabetes Mellitus'].replace(to_replace={'yes
data['Coronary Artery Disease'] = data['Coronary Artery Disease'].replace(to_r
data['Appetite'] = data['Appetite'].replace(to_replace={'good':1,'poor':0})
data['Pedal Edema'] = data['Pedal Edema'].replace(to_replace={'yes':1,'no':0})
data['Anemia'] = data['Anemia'].replace(to_replace={'yes':1,'no':0})

data['Red Blood Cells'] = data['Red Blood Cells'].replace(to_replace={'abnorma
data['Pus Cell'] = data['Pus Cell'].replace(to_replace={'abnormal':1,'normal'

data['Pus Cell clumps'] = data['Pus Cell clumps'].replace(to_replace={'present
data['Bacteria'] = data['Bacteria'].replace(to_replace={'present':1,'notpresen

data['Class'] = data['Class'].replace(to_replace={'ckd':1.0,'ckd\t':1.0,'notck

data.to_csv("Out1.csv")



data['Pedal Edema'] = data['Pedal Edema'].replace(to_replace='good',value=0) #
data['Appetite'] = data['Appetite'].replace(to_replace='no',value=0)
data['Coronary Artery Disease'] = data['Coronary Artery Disease'].replace(to_r
data['Diabetes Mellitus'] = data['Diabetes Mellitus'].replace(to_replace={'\tr



data1=data.dropna(axis=0)
data1.shape
```

```
(158, 25)
```

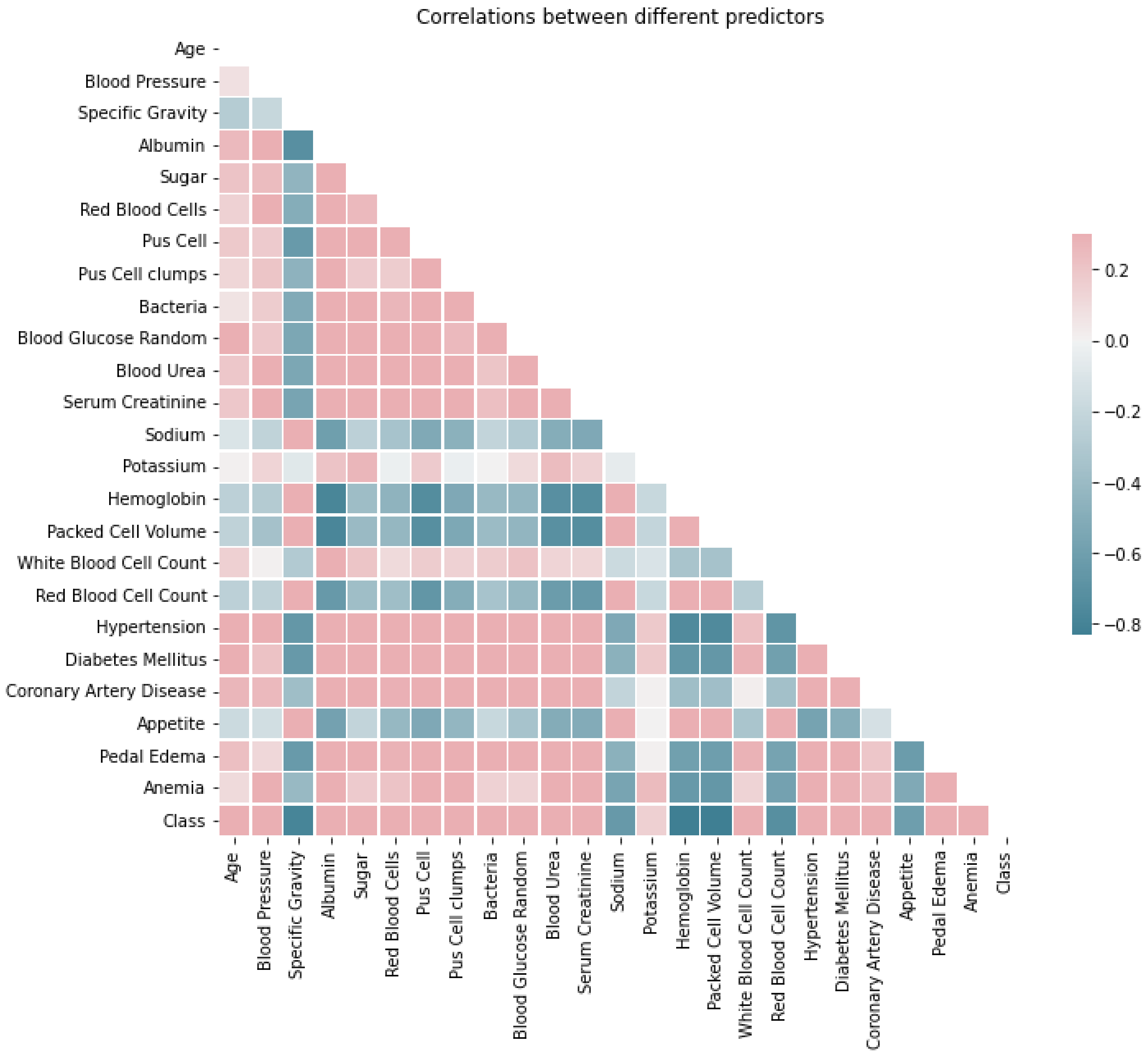
```
data1.isna().sum().sort_values(ascending=False)
```

```
Age                        0
Potassium                  0
Anemia                     0
Pedal Edema                0
Appetite                   0
Coronary Artery Disease    0
Diabetes Mellitus          0
Hypertension               0
Red Blood Cell Count       0
White Blood Cell Count     0
Packed Cell Volume         0
Hemoglobin                 0
Sodium                     0
Blood Pressure             0
Serum Creatinine           0
Blood Urea                 0
Blood Glucose Random       0
Bacteria                   0
Pus Cell clumps            0
```

```
Pus Cell                 0
Red Blood Cells          0
Sugar                    0
Albumin                  0
Specific Gravity         0
Class                    0
dtype: int64
```

```
corr_df = data1.corr()
mask = np.zeros_like(corr_df, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr_df, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Correlations between different predictors')
plt.show()
```

```
<ipython-input-7-27a1ceaba938>:2: DeprecationWarning: `np.bool` is a deprecated alias for the
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.2
  mask = np.zeros_like(corr_df, dtype=np.bool)
```



KNN

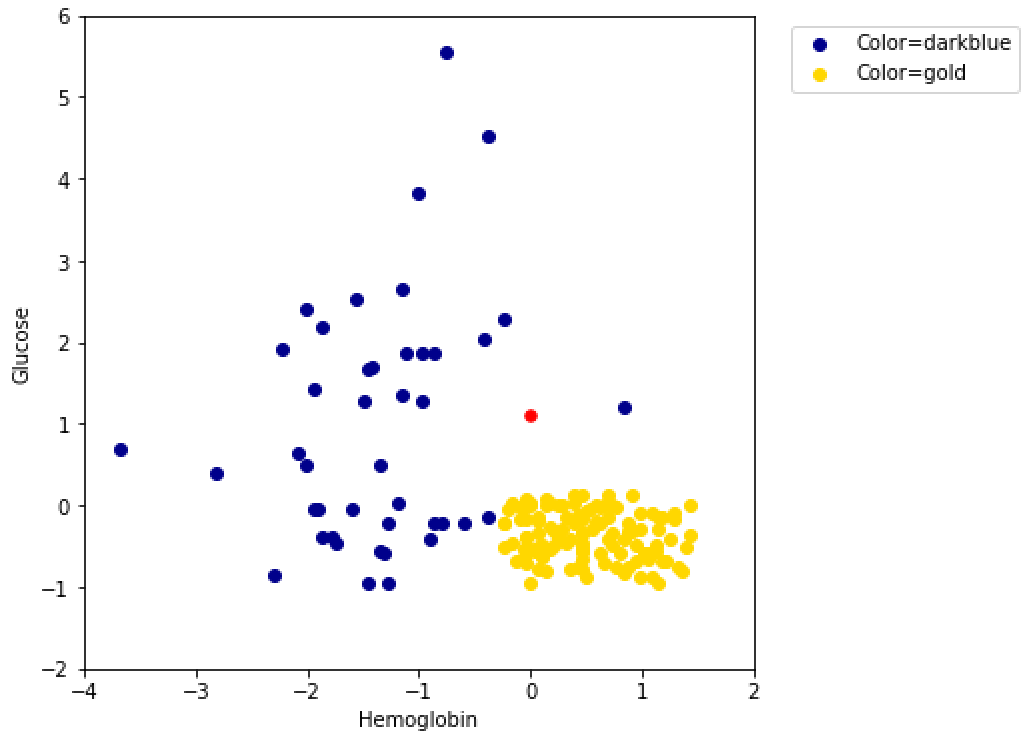
```
def standard_units(x):
    return (x - np.mean(x))/np.std(x)

ckd = pd.DataFrame(
    {'Hemoglobin':standard_units(data1['Hemoglobin']),
    'Glucose':standard_units(data1['Blood Glucose Random']),
    'Class':data1['Class']}
)

color_table = pd.DataFrame(
    {'Class':np.array([1, 0]),
    'Color':np.array(['darkblue', 'gold'])}
)

ckd = pd.merge(ckd, color_table, on='Class')

ckd
```

	Hemoglobin	Glucose	Class	Color
0	-0.865744	-0.221549	1.0	darkblue
1	-1.457446	-0.947597	1.0	darkblue
2	-1.004968	3.841231	1.0	darkblue
3	-2.814879	0.396364	1.0	darkblue
4	-2.083954	0.643529	1.0	darkblue
...	...	...	...	...
153	0.700526	0.133751	0.0	gold
154	0.978974	-0.870358	0.0	gold
155	0.735332	-0.484162	0.0	gold
156	0.178436	-0.267893	0.0	gold
157	0.735332	-0.005280	0.0	gold

158 rows × 4 columns

```
#Alice in Scatter plot
alice = np.array([0, 1.1])

ckd_darkblue = ckd[ckd['Color'] == 'darkblue']
ckd_gold = ckd[ckd['Color'] == 'gold']


fig, ax = plt.subplots(figsize=(6,6))
```

```
ax.scatter(ckd_darkblue['Hemoglobin'],
           ckd_darkblue['Glucose'],
           label='Color=darkblue',
           color='darkblue')
ax.scatter(ckd_gold['Hemoglobin'],
           ckd_gold['Glucose'],
           label='Color=gold',
           color='gold')
ax.scatter(alice[0],
           alice[1],
           color='red',
           s=30)


y_vals = ax.get_yticks()
plt.ylabel('Glucose')
ax.legend(bbox_to_anchor=(1.04,1), loc="upper left")
plt.xlabel('Hemoglobin')
plt.xlim(-4, 2)
plt.ylim(-2, 6);
plt.show()
```




```
#Euclidean Distance
def distance(point1, point2):
    return np.sqrt(np.sum((point1 - point2)**2))


def distance_from_alice(row):
    return distance(alice, np.array(row))
```

```
ckd1=ckd[['Hemoglobin','Glucose']]
ckd1
```

	Hemoglobin	Glucose
0	-0.865744	-0.221549
1	-1.457446	-0.947597
2	-1.004968	3.841231
3	-2.814879	0.396364
4	-2.083954	0.643529
...	...	...
153	0.700526	0.133751
154	0.978974	-0.870358
155	0.735332	-0.484162
156	0.178436	-0.267893
157	0.735332	-0.005280

158 rows × 2 columns

```
distances = ckd1.apply(distance_from_alice, axis=1)
ckd_with_distances = ckd.copy()
ckd_with_distances['Distance from Alice'] = distances
```

```
ckd_with_distances
```

	Hemoglobin	Glucose	Class	Color	Distance from Alice
0	-0.865744	-0.221549	1.0	darkblue	1.579875
1	-1.457446	-0.947597	1.0	darkblue	2.513325
2	-1.004968	3.841231	1.0	darkblue	2.919641
3	-2.814879	0.396364	1.0	darkblue	2.901491
4	-2.083954	0.643529	1.0	darkblue	2.133361
...	...	...	...	...	...
153	0.700526	0.133751	0.0	gold	1.193471
154	0.978974	-0.870358	0.0	gold	2.200159
155	0.735332	-0.484162	0.0	gold	1.746506
156	0.178436	-0.267893	0.0	gold	1.379482
157	0.735332	-0.005280	0.0	gold	1.327537

158 rows × 5 columns

```
sorted_by_distance = ckd_with_distances.sort_values(by=['Distance from Alice'
sorted_by_distance
```

	Hemoglobin	Glucose	Class	Color	Distance from Alice
14	0.839750	1.215099	1.0	darkblue	0.847601
35	-0.970162	1.276890	1.0	darkblue	0.986156
84	-0.030400	0.087407	0.0	gold	1.013049
152	0.143630	0.087407	0.0	gold	1.022728
6	-0.413266	2.049282	1.0	darkblue	1.035338
...	...	...	...	...	...
2	-1.004968	3.841231	1.0	darkblue	2.919641
12	-2.292790	-0.854910	1.0	darkblue	3.013065
41	-0.378460	4.520935	1.0	darkblue	3.441806
42	-3.685029	0.689873	1.0	darkblue	3.707782
36	-0.761326	5.540492	1.0	darkblue	4.505285

158 rows × 5 columns

```
alice_5_nearest_neighbors = sorted_by_distance.take(np.arange(5))
alice_5_nearest_neighbors
```

	Hemoglobin	Glucose	Class	Color	Distance from Alice
14	0.839750	1.215099	1.0	darkblue	0.847601
35	-0.970162	1.276890	1.0	darkblue	0.986156
84	-0.030400	0.087407	0.0	gold	1.013049
152	0.143630	0.087407	0.0	gold	1.022728
6	-0.413266	2.049282	1.0	darkblue	1.035338

```
a=alice_5_nearest_neighbors.groupby('Class').count()
```

```
nothaving_ckd=a.iloc[0,3]
nothaving_ckd
```

2

```
Having_ckd=a.iloc[1,3]
Having_ckd
```

3

```
if nothaving_ckd>Having_ckd:
  print("Alice does not have Chronic Kidney disease")
```

```
else:
  print("Alice has Chronic Kidney disease")
```

```
Alice has Chronic Kidney disease
```

RandomForest

```
data1.head()
```

	Age	Blood Pressure	Specific Gravity	Albumin	Sugar	Red Blood Cells	Pus Cell	Pus Cell clumps	Bacteria	Blood Glucose Random	...	Packe Cel Volur
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	32
9	53.0	90.0	1.020	2.0	0.0	1.0	1.0	1.0	0.0	70.0	...	29
11	63.0	70.0	1.010	3.0	0.0	1.0	1.0	1.0	0.0	380.0	...	32
14	68.0	80.0	1.010	3.0	2.0	0.0	1.0	1.0	1.0	157.0	...	16
20	61.0	80.0	1.015	2.0	0.0	1.0	1.0	0.0	0.0	173.0	...	24

5 rows × 25 columns

```
ckd.head()
```

	Hemoglobin	Glucose	Class	Color
0	-0.865744	-0.221549	1.0	darkblue
1	-1.457446	-0.947597	1.0	darkblue
2	-1.004968	3.841231	1.0	darkblue
3	-2.814879	0.396364	1.0	darkblue
4	-2.083954	0.643529	1.0	darkblue

```
ckd_copy=ckd.copy()
ckd_copy=ckd_copy.drop('Color',1)
```

```
<ipython-input-24-31e986d8ffe8>:2: FutureWarning: In a future version of pandas all arguments
  ckd_copy=ckd_copy.drop('Color',1)
```

```
x=ckd_copy.drop('Class',axis=1)
y=ckd_copy['Class']
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
print("X_train size {} , X_test size {}".format(X_train.shape,X_test.shape))
```

```
X_train size (126, 2) , X_test size (32, 2)
```

```
score=cross_val_score(RandomForestClassifier(max_depth=15,n_estimators=5),X_tr
print("Average Accuracy Score {}".format(score.mean()))
```

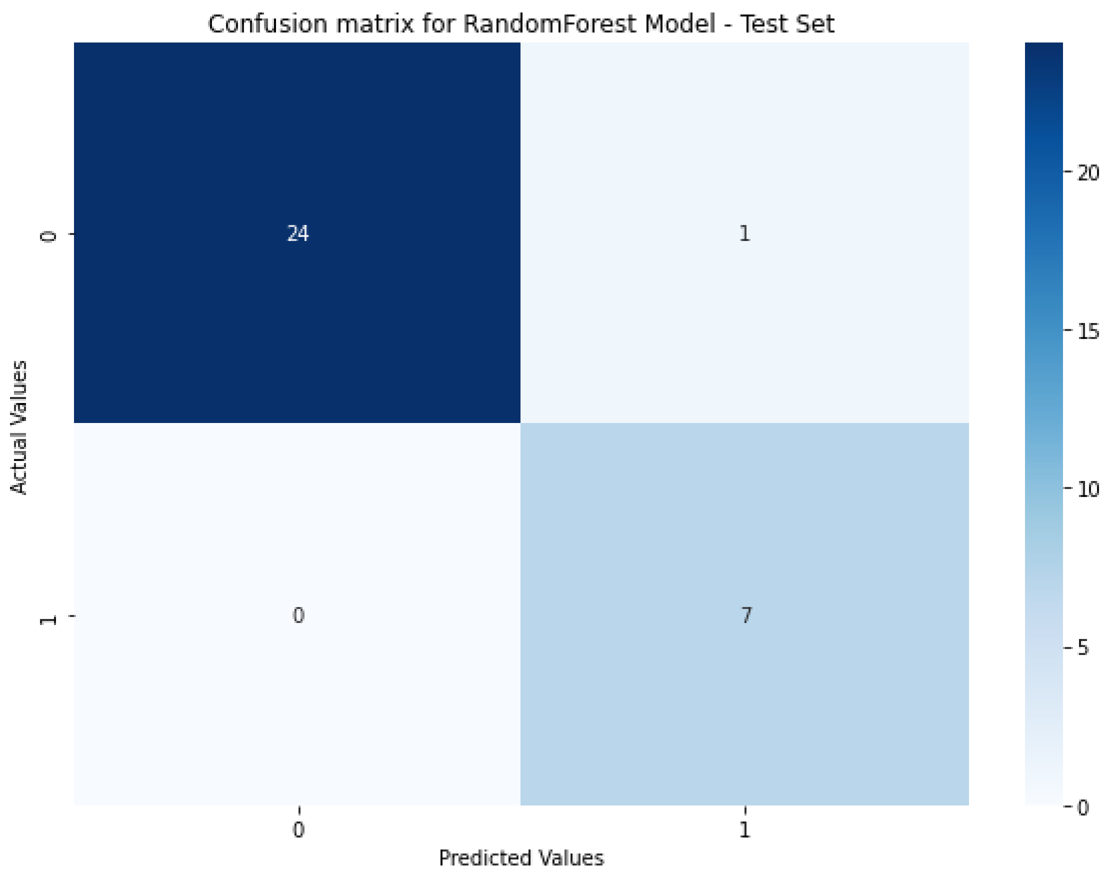
```
Average Accuracy Score 0.9923076923076923
```

```
rf=RandomForestClassifier(max_depth=5,n_estimators=5)
rf.fit(X_train,y_train)
```

```
RandomForestClassifier(max_depth=5, n_estimators=5)
```

```
y_pred=rf.predict(X_test)
confusionmatrix=confusion_matrix(y_pred,y_test)
```

```
plt.figure(figsize=(10,7))
p = sns.heatmap(confusionmatrix, annot=True, cmap="Blues", fmt='g')
plt.title('Confusion matrix for RandomForest Model - Test Set')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.show()
```




```
score=round(accuracy_score(y_test,y_pred),3)
print("Accuracy on the Test set: {}".format(score))
```

```
Accuracy on the Test set: 0.969
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.96      1.00      0.98        24
         1.0       1.00      0.88      0.93         8

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32
```

```
X_train=X_train[['Hemoglobin','Glucose']]
X_test=X_test[['Hemoglobin','Glucose']]
rf.fit(X_train,y_train)
def predict(hemo,gl):
    x=[[hemo,gl]]
    return rf.predict(x)
```

```
prediction=predict(0,1.1)[0]
prediction
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid
  warnings.warn(
1.0
```

```
if prediction:
  print('Oops! You have Chronic Kidney Disease.')
else:
  print("Great! You don't have Chronic Kidney Disease.")
```

```
Oops! You have Chronic Kidney Disease.
```

DecisionTree

```
ckd_copy2=ckd_copy.copy()
ckd_copy2
```

	Hemoglobin	Glucose	Class
0	-0.865744	-0.221549	1.0
1	-1.457446	-0.947597	1.0
2	-1.004968	3.841231	1.0
3	-2.814879	0.396364	1.0

```
dtc = tree.DecisionTreeClassifier()
model = dtc.fit(X_train, y_train)
```

153	0.700526	0.133751	0.0

```
X_train
```

	Hemoglobin	Glucose
16	-1.561864	2.528165
130	0.874556	-0.746776
134	0.909362	-0.314236
22	-2.014342	2.420030
93	0.074018	-0.345132
...	...	...
9	-1.353028	0.489051
103	1.431451	0.010168
67	0.282854	-0.298788
117	0.456884	-0.561402
47	-0.239236	-0.499610

126 rows × 2 columns

```
predict_train = model.predict(X_train)
predict_train
```

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

```
predict_test = model.predict(X_test)
predict_test
```

```
array([ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  1.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  1.,  1.,  1.,
```

```
         0.,  1.,  0.,  0.,  0.,  0.])
```

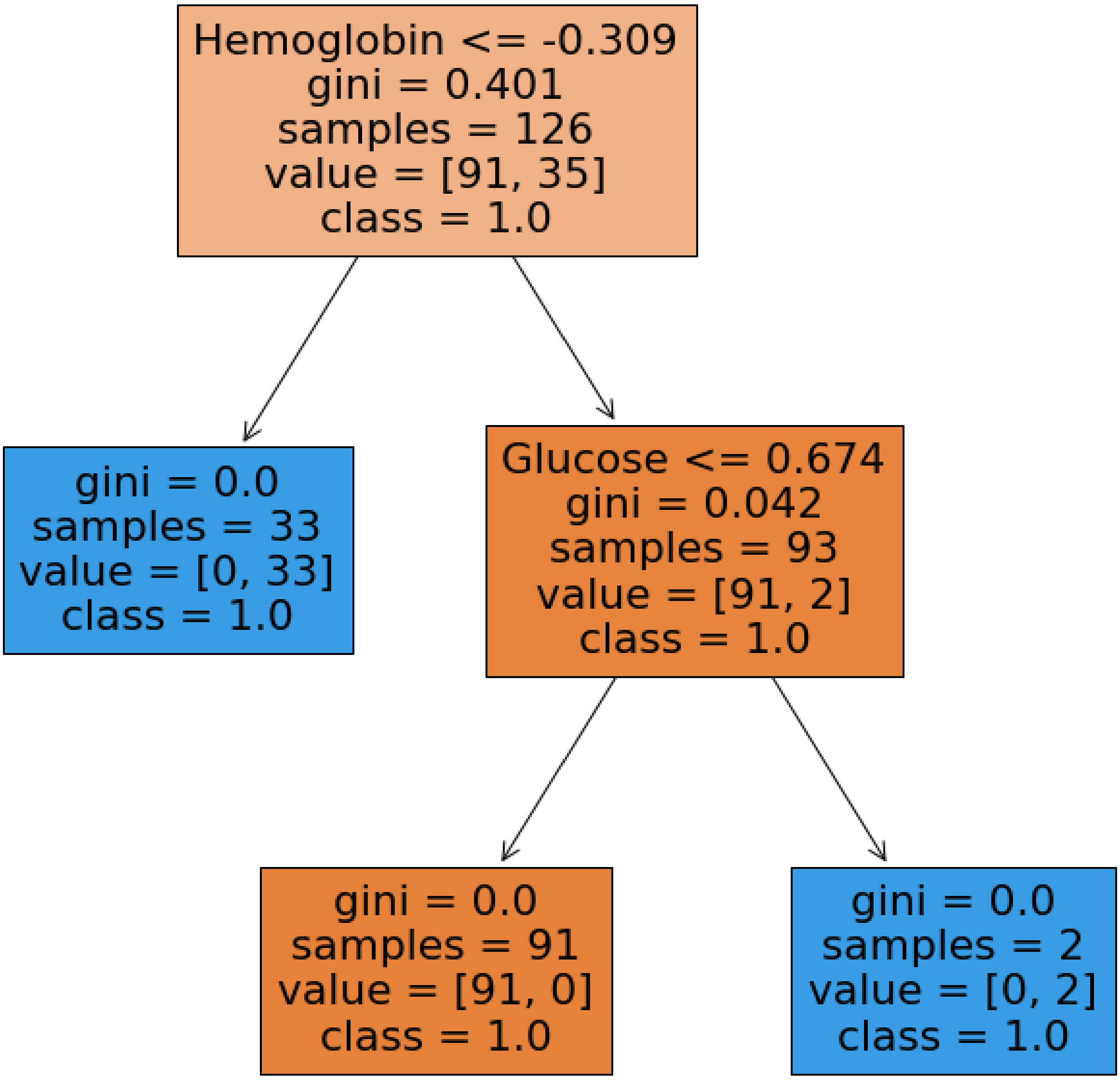
```python
dtc_acc = accuracy_score(y_test, dtc.predict(X_test))
print("\n")
print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_tra
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc} \n")
print("\n")
print(f"Confusion Matrix :- \n{confusion_matrix(y_test, dtc.predict(X_test))}
confusion = confusion_matrix(y_test, dtc.predict(X_test))
tn, fp, fn, tp = confusion.ravel()
print("\n")

print("TN -->",tn)
print("FP -->",fp)
print("FN -->",fn)
print("TP -->",tp)
print("\n")
print("\n")

print(f"Classification Report :- \n {classification_report(y_test, dtc.predic
```

```
Training Accuracy of Decision Tree Classifier is 1.0
Test Accuracy of Decision Tree Classifier is 1.0



Confusion Matrix :-
[[24  0]
 [ 0  8]]



TN --> 24
FP --> 0
FN --> 0
TP --> 8




Classification Report :-
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        24
         1.0       1.00      1.00      1.00         8

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32
```

```python
ind_col=ckd_copy2[['Hemoglobin','Glucose']]
dep_col=ckd_copy2['Class']
```

```
ind_col = ind_col.astype(str)
dep_col = dep_col.astype(str)
feature_names = ind_col.columns.tolist()
plt.figure(figsize=(12, 12))
plot_tree(model, filled=True, feature_names=feature_names, class_names=dep_col
```

```
[Text(0.4, 0.833333333333, 'Hemoglobin <= -0.309\ngini = 0.401\nsamples = 126\nvalue = [91, 35
= 1.0'),
 Text(0.2, 0.5, 'gini = 0.0\nsamples = 33\nvalue = [0, 33]\nclass = 1.0'),
 Text(0.6, 0.5, 'Glucose <= 0.674\ngini = 0.042\nsamples = 93\nvalue = [91, 2]\nclass = 1.0'),
 Text(0.4, 0.166666666667, 'gini = 0.0\nsamples = 91\nvalue = [91, 0]\nclass = 1.0'),
 Text(0.8, 0.166666666667, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]\nclass = 1.0')]
```




```
Alice = np.array([[0, 1.1]])

final=model.predict(Alice)
final
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid
  warnings.warn(
```

```
array([ 1.])
```

```
if final==0:
  print("Alice does not have CKD")
else:
  print("Alice has CKD")
```

```
Alice has CKD
```

Q5.

Validation

```
wine=pd.read_csv("/content/winequality-red.csv")
wine.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56

```
wine.isna().any()
```

```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
dtype: bool
```

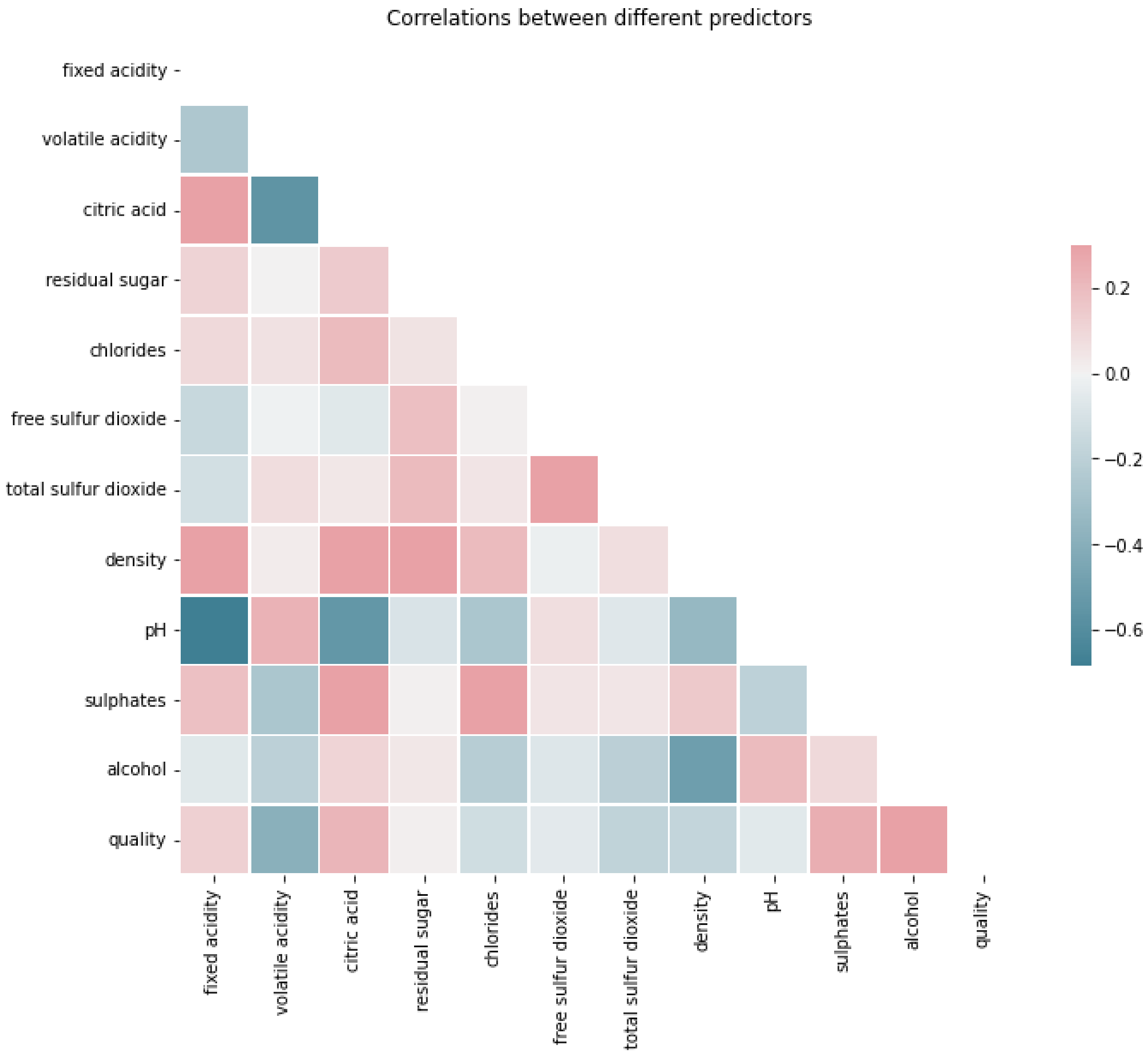
```
wine['quality'].value_counts()
```

```
5    681
6    638
7    199
```

```
4      53
8      18
3      10
Name: quality, dtype: int64
```

```
corr_df = wine.corr()
mask = np.zeros_like(corr_df, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr_df, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Correlations between different predictors')
plt.show()
```

```
<ipython-input-47-af5de3b719fe>:2: DeprecationWarning: `np.bool` is a deprecated alias for the
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.2
  mask = np.zeros_like(corr_df, dtype=np.bool)
```



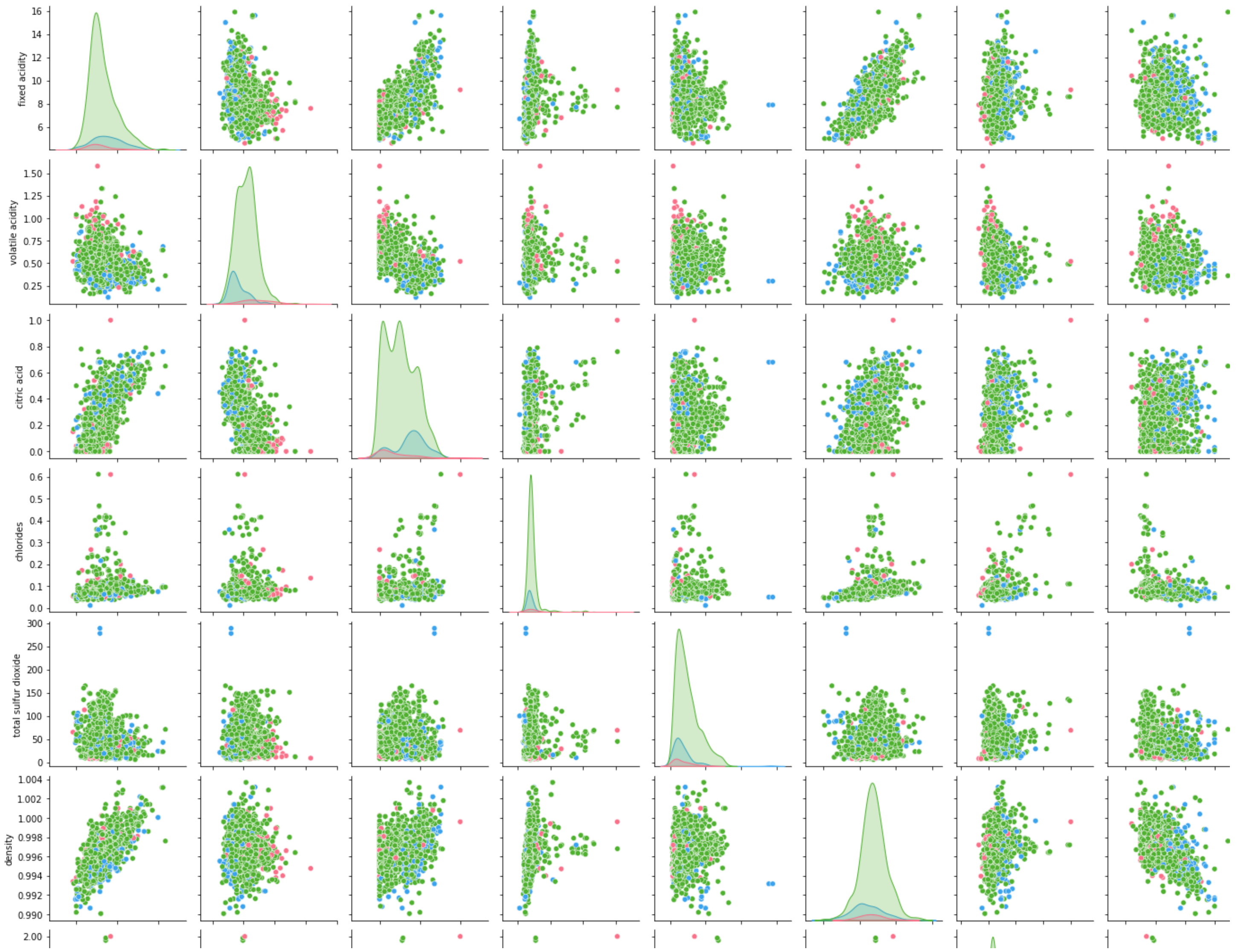

```
wine.drop(["residual sugar",'free sulfur dioxide','pH'],axis = 1,inplace = Tru
wine.head()
```

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	sulphates	alcohol
0	7.4	0.70	0.00	0.076	34.0	0.9978	0.56	9.4
1	7.8	0.88	0.00	0.098	67.0	0.9968	0.68	9.8
2	7.8	0.76	0.04	0.092	54.0	0.9970	0.65	9.8
3	11.2	0.28	0.56	0.075	60.0	0.9980	0.58	9.8
4	7.4	0.70	0.00	0.076	34.0	0.9978	0.56	9.4

```
bins = [0, 4, 6, 10]
labels = ["poor","normal","excellent"]
wine['quality_label'] = pd.cut(wine['quality'], bins=bins, labels=labels)
wine.drop('quality',axis =1, inplace = True)
wine.head()
```

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	sulphates	alcohol	qualit
0	7.4	0.70	0.00	0.076	34.0	0.9978	0.56	9.4	
1	7.8	0.88	0.00	0.098	67.0	0.9968	0.68	9.8	
2	7.8	0.76	0.04	0.092	54.0	0.9970	0.65	9.8	
3	11.2	0.28	0.56	0.075	60.0	0.9980	0.58	9.8	

```
sns.pairplot(wine, hue="quality_label", palette="husl",diag_kind="kde")
plt.show()
```




```
#df_wine = pd.get_dummies(wine, columns=['alcohol_label'], drop_first=True)
#df_wine.head()
```

```
wine.head()
```

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	sulphates	alcohol	qualit
0	7.4	0.70	0.00	0.076	34.0	0.9978	0.56	9.4	
1	7.8	0.88	0.00	0.098	67.0	0.9968	0.68	9.8	
2	7.8	0.76	0.04	0.092	54.0	0.9970	0.65	9.8	
3	11.2	0.28	0.56	0.075	60.0	0.9980	0.58	9.8	

```
wine_label=wine['quality_label']
#wine_rest=wine.drop(['alcohol_label','quality_label'],1)
wine_rest=wine.drop(['quality_label'],1)
```

```
<ipython-input-53-6eaaa66e9f66>:3: FutureWarning: In a future version of pandas all arguments
  wine_rest=wine.drop(['quality_label'],1)
```

```
wine_label
```

```
0       normal
1       normal
2       normal
3       normal
4       normal
         ...
1594    normal
1595    normal
1596    normal
1597    normal
1598    normal
Name: quality_label, Length: 1599, dtype: category
Categories (3, object): ['poor' < 'normal' < 'excellent']
```

```
wine_rest
```

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	sulphates
0	7.4	0.700	0.00	0.076	34.0	0.99780	0.56
1	7.8	0.880	0.00	0.098	67.0	0.99680	0.68
2	7.8	0.760	0.04	0.092	54.0	0.99700	0.65
3	11.2	0.280	0.56	0.075	60.0	0.99800	0.58
4	7.4	0.700	0.00	0.076	34.0	0.99780	0.56
...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	0.090	44.0	0.99490	0.58
1595	5.9	0.550	0.10	0.062	51.0	0.99512	0.76
1596	6.3	0.510	0.13	0.076	40.0	0.99574	0.75
1597	5.9	0.645	0.12	0.075	44.0	0.99547	0.71
1598	6.0	0.310	0.47	0.067	42.0	0.99549	0.66

1599 rows × 8 columns

KNN

```
X_train, X_test, Y_train, Y_test = train_test_split(wine_rest, wine_label, tes
```

```
X_train
```

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	sulphates
730	9.5	0.550	0.66	0.387	37.0	0.99820	0.67
932	7.6	0.400	0.29	0.078	66.0	0.99710	0.59
821	4.9	0.420	0.00	0.048	42.0	0.99154	0.74
985	7.4	0.580	0.00	0.064	11.0	0.99562	0.58
549	9.0	0.530	0.49	0.171	25.0	0.99750	0.61
...	...	...	...	...	...	...	...

```
X_test
```

	fixed acidity	volatile acidity	citric acid	chlorides	total sulfur dioxide	density	sulphates
1429	7.9	0.180	0.40	0.049	67.0	0.99600	0.93
260	7.9	0.330	0.23	0.077	45.0	0.99625	0.65
916	5.3	0.715	0.19	0.161	62.0	0.99395	0.61
1141	8.2	0.380	0.32	0.080	71.0	0.99624	0.85
1574	5.6	0.310	0.78	0.074	92.0	0.99677	0.48
...	...	...	...	...	...	...	...
298	7.2	0.650	0.02	0.094	31.0	0.99930	0.80
571	6.2	0.360	0.24	0.095	42.0	0.99460	0.57
605	8.3	0.600	0.13	0.085	24.0	0.99840	0.59
1548	11.2	0.400	0.50	0.099	50.0	0.99783	0.58
455	11.3	0.620	0.67	0.086	19.0	0.99880	0.69

480 rows × 8 columns

```
scaler = StandardScaler()
scaler.fit(wine_rest)
scaled_features = scaler.transform(wine_rest)
wine_rest_scaled= pd.DataFrame(scaled_features, columns=wine_rest.columns)


X_train_sc, X_test_sc, y_train_sc, y_test_sc = train_test_split(wine_rest_scal


X_train_sc = X_train_sc.to_numpy()
y_train_sc = y_train_sc.to_numpy()


def apply_knn(neigh, weight='uniform'):
    knn = KNeighborsClassifier(n_neighbors=neigh, weights=weight)
    knn.fit(X_train_sc,y_train_sc)
    pred_knn = knn.predict(X_test_sc)
    return pred_knn
```

```
model = KNeighborsClassifier()

params = {'n_neighbors':list(range(1, 50, 2)), 'weights':['uniform', 'distanc

gs = GridSearchCV(model, params, cv = 5, n_jobs=-1)

gs_results = gs.fit(X_train_sc, y_train_sc)
```

```
print('Best Accuracy: ', gs_results.best_score_)
print('Best Parametrs: ', gs_results.best_params_)
```

```
Best Accuracy:  0.852570467649
Best Parametrs:  {'n_neighbors': 9, 'weights': 'distance'}
```

```
pred_knn = apply_knn(9)
print('Accuracy of model at K=9 is', accuracy_score(y_test_sc, pred_knn))
```

```
Accuracy of model at K=9 is 0.84375
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:443: UserWarning: X has feature names,
  warnings.warn(
```

RandomForest

```
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train_sc, y_train_sc)
pred_rfc = rfc.predict(X_test_sc)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:443: UserWarning: X has feature names,
  warnings.warn(
```

```
y_test_sc.shape
```

```
(480,)
```

```
pred_rfc.shape
```

```
(480,)
```

```
print(classification_report(y_test_sc, pred_rfc))
```

```
              precision    recall  f1-score   support

   excellent       0.82      0.55      0.66        65
      normal       0.89      0.98      0.93       395
        poor       0.00      0.00      0.00        20
```

```
    accuracy                           0.88       480
   macro avg       0.57      0.51      0.53       480
weighted avg       0.84      0.88      0.85       480
```

```
print(confusion_matrix(y_test_sc, pred_rfc))
```

```
[[ 36  29   0]
 [  8 386   1]
 [  0  20   0]]
```

```
# the Accuracy of Random Forest is around 84%
```

Support Vector Classifier

```
svc = SVC()
svc.fit(X_train_sc, y_train_sc)
pred_svc = svc.predict(X_test_sc)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:443: UserWarning: X has feature names,
  warnings.warn(
```

```
print(classification_report(y_test_sc, pred_svc))
```

```
              precision    recall  f1-score   support

   excellent       0.89      0.25      0.39        65
      normal       0.85      0.99      0.92       395
        poor       0.00      0.00      0.00        20

    accuracy                           0.85       480
   macro avg       0.58      0.41      0.43       480
weighted avg       0.82      0.85      0.81       480

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetri
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetri
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetri
  _warn_prf(average, modifier, msg_start, len(result))
```

```
#Support vector classifier gets 79%
```

PART-D

Q6

```
house=pd.read_csv("/content/House.csv")
```

```
house.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating
0	13300000	7420	4	2	3	yes	no	no	no
1	12250000	8960	4	4	4	yes	no	no	no
2	12250000	9960	3	2	2	yes	no	yes	no
3	12215000	7500	4	2	2	yes	no	yes	no
4	11410000	7420	4	1	2	yes	yes	yes	no

```
house.shape
```

```
(545, 13)
```

```
house.describe()
```

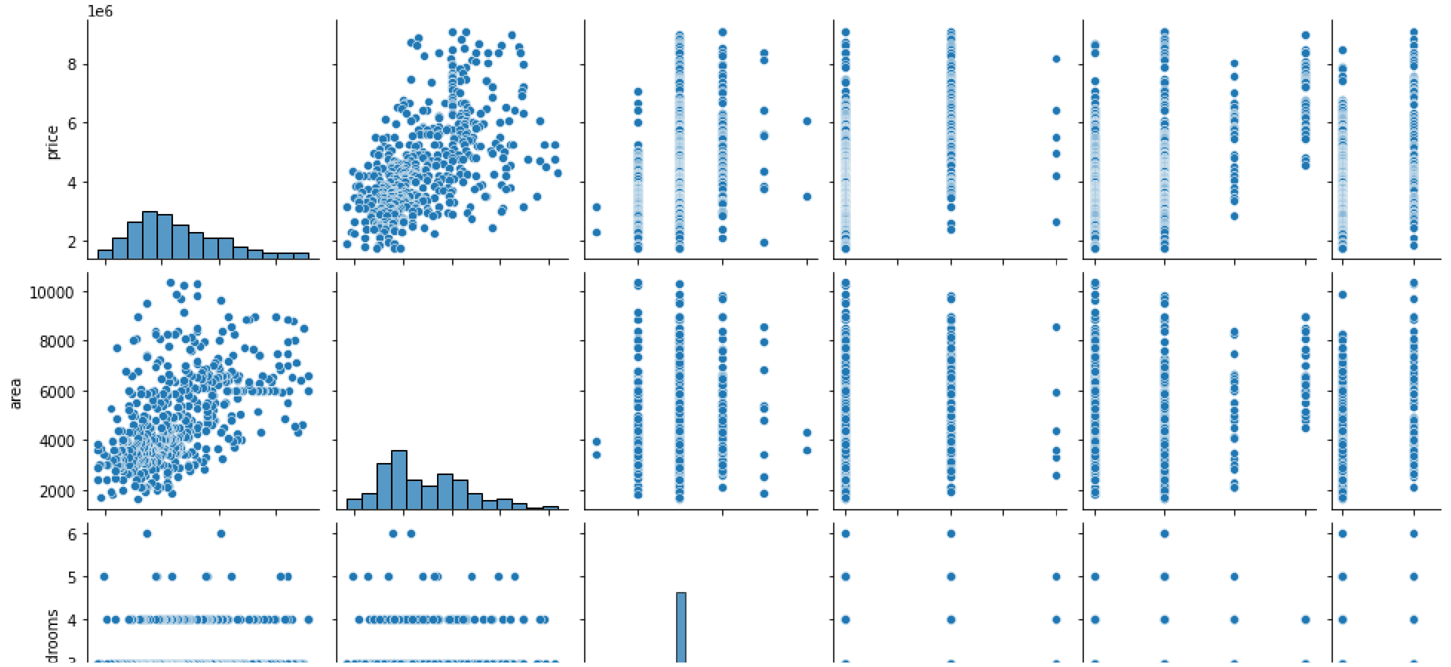
	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

```
house.isna().any()
```

```
price               False
area                False
bedrooms            False
bathrooms           False
stories             False
mainroad            False
guestroom           False
basement            False
hotwaterheating     False
airconditioning     False
parking             False
prefarea            False
```

```
furnishingstatus    False
dtype: bool
```

```
Q1 = house.price.quantile(0.25)
Q3 = house.price.quantile(0.75)
IQR = Q3 - Q1
house = house[(house.price >= Q1 - 1.5*IQR) & (house.price <= Q3 + 1.5*IQR)]

Q1 = house.area.quantile(0.25)
Q3 = house.area.quantile(0.75)
IQR = Q3 - Q1
house = house[(house.area >= Q1 - 1.5*IQR) & (house.area <= Q3 + 1.5*IQR)]

sns.pairplot(house)
plt.show()
```




```
varlist =  ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'aircondi

def binary_map(x):
    return x.map({'yes': 1, "no": 0})

house[varlist] = house[varlist].apply(binary_map)
```

```
house.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheatin
15	9100000	6000	4	1	2	1	0	1	(
16	9100000	6600	4	2	2	1	1	1	(
17	8960000	8500	3	2	4	1	0	0	(
18	8890000	4600	3	2	2	1	1	0	(
19	8855000	6420	3	2	2	1	0	0	(

```
status = pd.get_dummies(house['furnishingstatus'])

status.head()
```

furnished	semi-furnished	unfurnished

```
status = pd.get_dummies(house['furnishingstatus'], drop_first = True)
house = pd.concat([house, status], axis = 1)

house.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheatin
15	9100000	6000	4	1	2	1	0	1	(
16	9100000	6600	4	2	2	1	1	1	(
17	8960000	8500	3	2	4	1	0	0	(
18	8890000	4600	3	2	2	1	1	0	(
19	8855000	6420	3	2	2	1	0	0	(

```
house.drop(['furnishingstatus'], axis = 1, inplace = True)
house.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheatin
15	9100000	6000	4	1	2	1	0	1	(
16	9100000	6600	4	2	2	1	1	1	(
17	8960000	8500	3	2	4	1	0	0	(
18	8890000	4600	3	2	2	1	1	0	(
19	8855000	6420	3	2	2	1	0	0	(

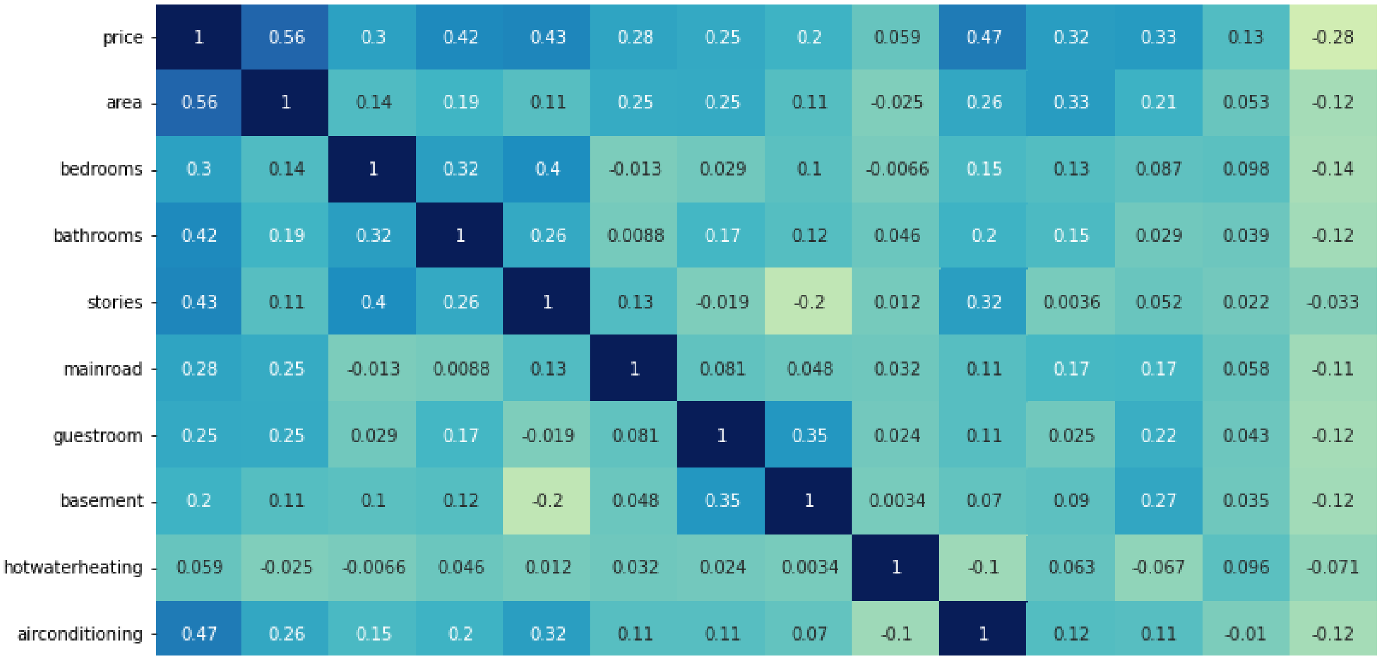
```
np.random.seed(0)
df_train, df_test = train_test_split(house, train_size = 0.7, test_size = 0.3

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
num_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking','price']
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
df_train.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterh
148	0.523810	0.526907	0.4	0.0	0.666667	1	0	0	
236	0.390476	0.114134	0.2	0.0	0.333333	1	1	1	
356	0.275238	0.072738	0.8	0.5	0.000000	0	0	1	
425	0.219048	0.151390	0.2	0.0	0.000000	1	0	1	
516	0.095238	0.157895	0.2	0.0	0.000000	0	1	0	

```
df_train.describe()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basen
count	361.000000	361.000000	361.000000	361.000000	361.000000	361.000000	361.000000	361.00(
mean	0.383701	0.350081	0.390582	0.127424	0.268698	0.875346	0.168975	0.34(
std	0.209712	0.207184	0.149146	0.224465	0.287833	0.330784	0.375250	0.47
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
25%	0.237143	0.189829	0.200000	0.000000	0.000000	1.000000	0.000000	0.00(
50%	0.338095	0.295092	0.400000	0.000000	0.333333	1.000000	0.000000	0.00(
75%	0.514286	0.491425	0.400000	0.000000	0.333333	1.000000	0.000000	1.00(
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00(

```
plt.figure(figsize = (16, 10))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```




```
y_train = df_train.pop('price')
X_train = df_train
```

```
y_test = df_test.pop('price')
X_test = df_test
```

```
X_train = X_train.to_numpy()
y_train = y_train.to_numpy()

from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor(n_neighbors=6, weights='uniform')
knn.fit(X_train,y_train)
```

```
KNeighborsRegressor(n_neighbors=6)
```

```
def apply_knn(neigh, weight='uniform'):
    knn = KNeighborsRegressor(n_neighbors=neigh, weights=weight)
    knn.fit(X_train,y_train)
    pred_knn = knn.predict(X_test)
    return pred_knn

model = KNeighborsRegressor()
params = {'n_neighbors':list(range(1, 50, 2)), 'weights':['uniform', 'distanc
gs = GridSearchCV(model, params, cv = 5, n_jobs=-1)
gs_results = gs.fit(X_train, y_train)

print('Best Accuracy: ', gs_results.best_score_)
print('Best Parametrs: ', gs_results.best_params_)
```

```
Best Accuracy:  0.523383914449
Best Parametrs:  {'n_neighbors': 9, 'weights': 'distance'}
```

```
pred_knn = apply_knn(9)
from sklearn.metrics import mean_absolute_error
print("MAE",mean_absolute_error(y_test,pred_knn))
```

```
MAE 4565790.77872
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:443: UserWarning: X has feature names,
  warnings.warn(
```

```
from sklearn.metrics import mean_squared_error
print("RMSE",np.sqrt(mean_squared_error(y_test,pred_knn)))
```

```
RMSE 4876137.31684
```

```
print(knn.score(X_test,y_test))
```

```
-7.11417318899
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:443: UserWarning: X has feature names,
  warnings.warn(
```

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
```

```
regr.fit(X_train, y_train)
```

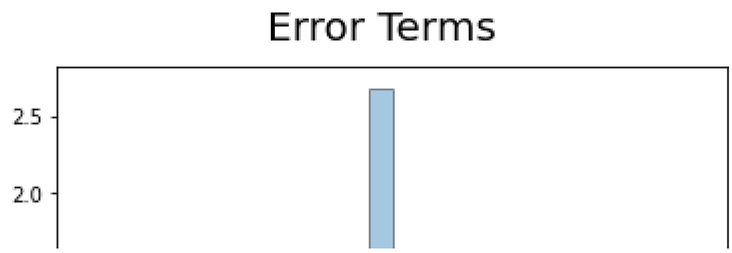
```
LinearRegression()
```

```
y_train_price = regr.predict(X_train)
```

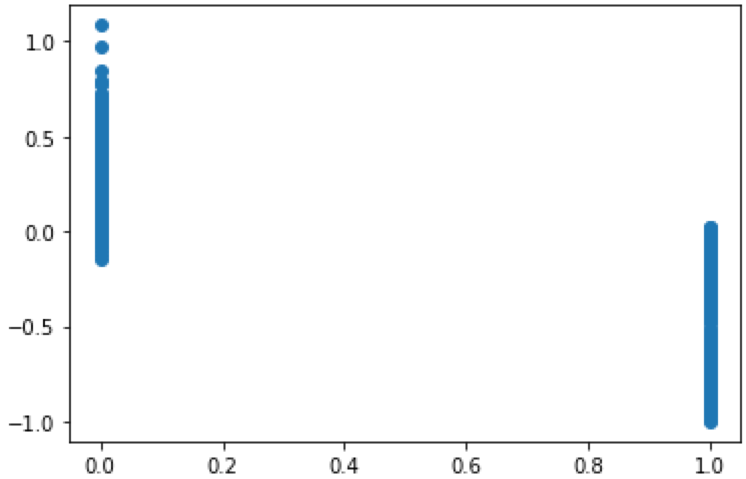
```
res = (y_train_price - y_train)
```

```
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)
plt.xlabel('Errors', fontsize = 18)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
Text(0.5, 0, 'Errors')
```




```
plt.scatter(y_train,res)
plt.show()
```



Q7

```
train=pd.read_csv("/content/train.csv")
```

```
test=pd.read_csv("/content/test(1).csv")
```

```
train.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
test.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
```

```
Sex             0
Age            86
SibSp           0
Parch           0
Ticket          0
Fare            1
Cabin         327
Embarked        0
dtype: int64
```

```
impute_value = train['Age'].median()

test['Age'] = test['Age'].fillna(impute_value)

train['Age'] = train['Age'].fillna(impute_value)

train['IsFemale'] = (train['Sex'] == 'female').astype(int)
test['IsFemale'] = (test['Sex'] == 'female').astype(int)

predictors = ['Pclass', 'IsFemale', 'Age']

X_train = train[predictors].values
X_train
```

```
array([[  3.,   0.,  22.],
       [  1.,   1.,  38.],
       [  3.,   1.,  26.],
       ...,
       [  3.,   1.,  28.],
       [  1.,   0.,  26.],
       [  3.,   0.,  32.]])
```

```
X_test = test[predictors].values
X_test
```

```
array([[  3. ,   0. ,  34.5],
       [  3. ,   1. ,  47. ],
       [  2. ,   0. ,  62. ],
       ...,
       [  3. ,   0. ,  38.5],
       [  3. ,   0. ,  28. ],
       [  3. ,   0. ,  28. ]])
```

```
y_train = train['Survived'].values
y_train
```

```
array([0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
```

```
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0])
```

```
X_train[:5]
```

```
array([[ 3.,  0., 22.],
       [ 1.,  1., 38.],
       [ 3.,  1., 26.],
       [ 1.,  1., 35.],
       [ 3.,  0., 35.]])
```

```
y_train[:5]
```

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

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
y_predict = model.predict(X_test)
y_predict[:10]
```

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

```
test
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	
...	...	...	...	...	...	...	...	...	...	...	
413	1305	3	Spector, Mr. Woolf	male	28.0	0	0	A.5. 3236	8.0500	NaN	

```
from sklearn.linear_model import LogisticRegressionCV
model_cv = LogisticRegressionCV()
model_cv.fit(X_train, y_train)
```

```
LogisticRegressionCV()
```

```
from sklearn.model_selection import cross_val_score
model = LogisticRegression(C=10)
scores = cross_val_score(model, X_train, y_train, cv=4)
scores
```

```
array([ 0.77578475,  0.79820628,  0.77578475,  0.78828829])
```