Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

Experiment No. 2
Analyze the Titanic Survival Dataset and apply appropriate regression technique.
Date of Performance: 17/08/23
Date of Submission: 24/08/23

Aim: Analyze the Titanic Survival Dataset and apply appropriate Regression Technique.

Objective: Able to perform various feature engineering tasks, apply logistic regression on the given dataset and maximize the accuracy.

Theory:

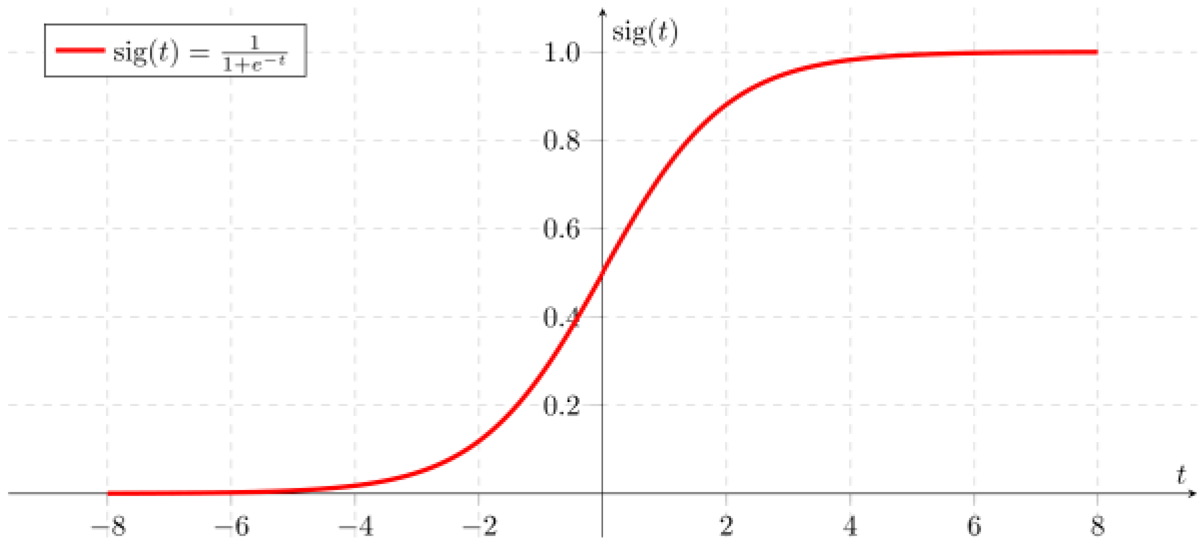
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification the logistic regression techniques makes use of Sigmoid function.

For example,

To predict whether an email is spam (1) or (0)

Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.
From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.



Dataset:

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

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

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

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

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

pclass: A proxy for socio-economic status (SES)

1st = Upper, 2nd = Middle, 3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...,

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored) parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

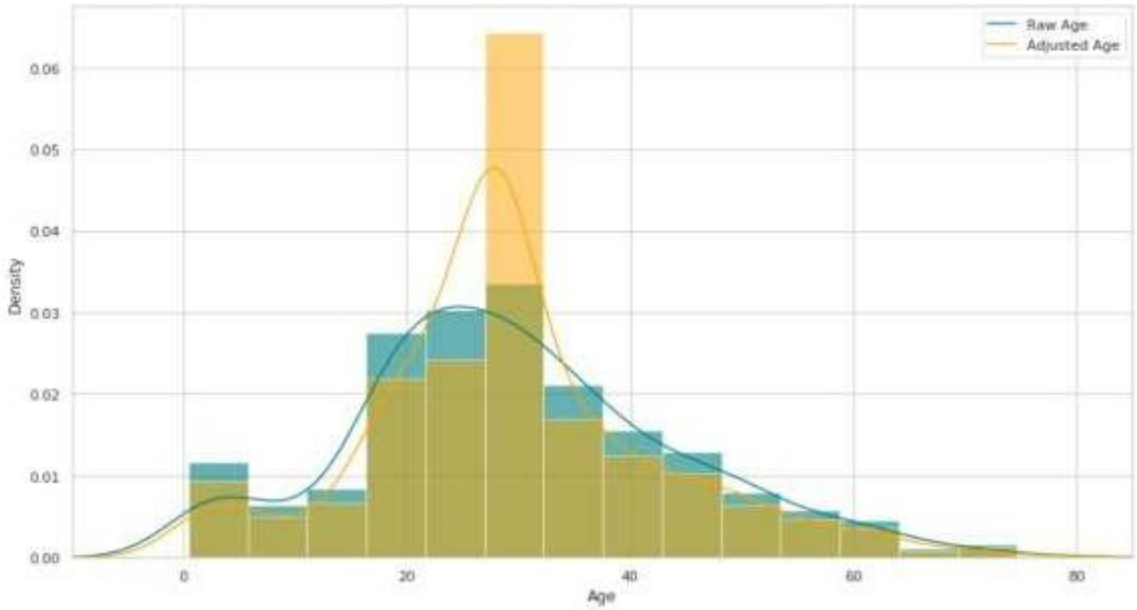
Some children travelled only with a nanny, therefore parch=0 for them.

Code:

```
import numpy as np

import pandas as pd

from sklearn import preprocessing

import matplotlib.pyplot as plt

plt.rc("font", size=14)

import seaborn as sns

sns.set(style="white")

sns.set(style="whitegrid", color_codes=True)

train_df = pd.read_csv("../input/train.csv")

test_df = pd.read_csv("../input/test.csv")

# Train data

train_df.head()
```

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

```
train_data = train_df.copy()

train_data["Age"].fillna(train_df["Age"].median(skipna=True), inplace=True)

train_data["Embarked"].fillna(train_df['Embarked'].value_counts().idxmax(),
inplace=True)

train_data.drop('Cabin', axis=1, inplace=True)

plt.figure(figsize=(15,8))

ax = train_df["Age"].hist(bins=15, density=True, stacked=True, color='teal',
alpha=0.6)

train_df["Age"].plot(kind='density', color='teal')

ax= train_data["Age"].hist(bins=15, density=True, stacked=True, color='orange',
alpha=0.5)

train_data["Age"].plot(kind='density', color='orange')

ax.legend(['Raw Age', 'Adjusted Age'])

ax.set(xlabel='Age')

plt.xlim(-10,85)

plt.show()
```



train_data['TravelAlone']=np.where((train_data["SibSp"]+train_data["Parch"])>
0, 0, 1)

train_data.drop('SibSp', axis=1, inplace=True)

train_data.drop('Parch', axis=1, inplace=True)

training=pd.get_dummies(train_data, columns=["Pclass","Embarked","Sex"])

training.drop('Sex_female', axis=1, inplace=True)

training.drop('PassengerId', axis=1, inplace=True)

training.drop('Name', axis=1, inplace=True)

training.drop('Ticket', axis=1, inplace=True)

final_train = training

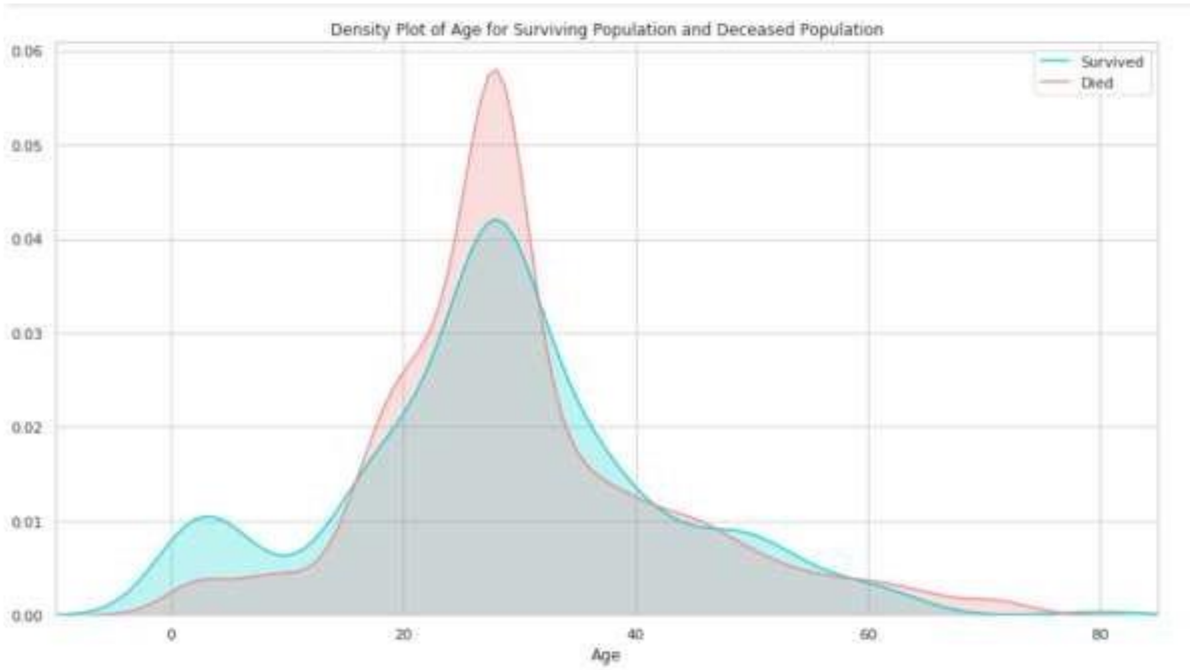
final_train.head()

	Survived	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S	Sex_male
0	0	22.0	7.2500	0	0	0	1	0	0	1	1
1	1	38.0	71.2833	0	1	0	0	1	0	0	0
2	1	26.0	7.9250	1	0	0	1	0	0	1	0
3	1	35.0	53.1000	0	1	0	0	0	0	1	0
4	0	35.0	8.0500	1	0	0	1	0	0	1	1

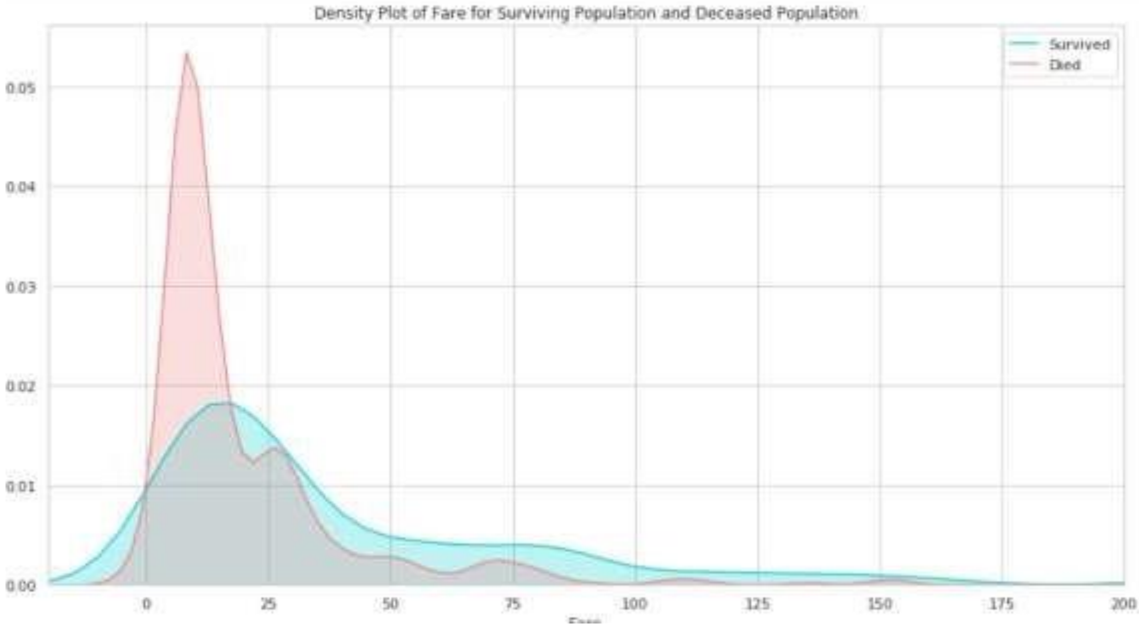
```
test_data = test_df.copy()

test_data["Age"].fillna(train_df["Age"].median(skipna=True), inplace=True)

test_data["Fare"].fillna(train_df["Fare"].median(skipna=True), inplace=True)

test_data.drop('Cabin', axis=1, inplace=True)

test_data['TravelAlone']=np.where((test_data["SibSp"]+test_data["Parch"])>0,
0, 1)

test_data.drop('SibSp', axis=1, inplace=True)

test_data.drop('Parch', axis=1, inplace=True)

testing = pd.get_dummies(test_data, columns=["Pclass","Embarked","Sex"])

testing.drop('Sex_female', axis=1, inplace=True)

testing.drop('PassengerId', axis=1, inplace=True)

testing.drop('Name', axis=1, inplace=True)

testing.drop('Ticket', axis=1, inplace=True)

final_test = testing

final_test.head()
```

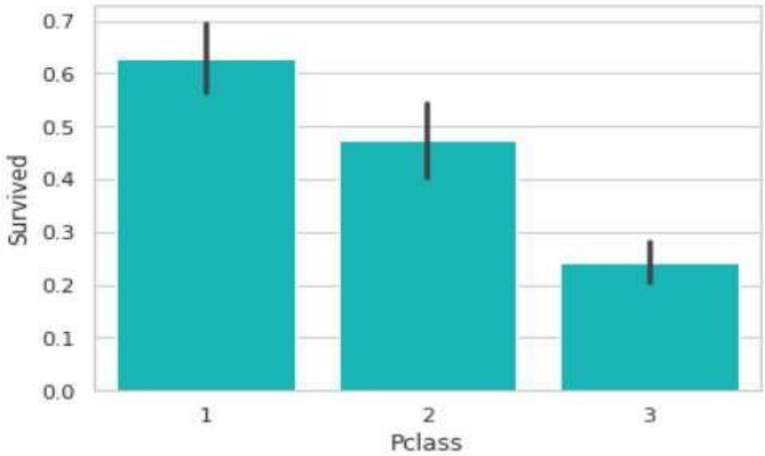
	Survived	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S	Sex_male
0	0	22.0	7.2500	0	0	0	1	0	0	1	1
1	1	38.0	71.2833	0	1	0	0	1	0	0	0
2	1	26.0	7.9250	1	0	0	1	0	0	1	0
3	1	35.0	53.1000	0	1	0	0	0	0	1	0
4	0	35.0	8.0500	1	0	0	1	0	0	1	1

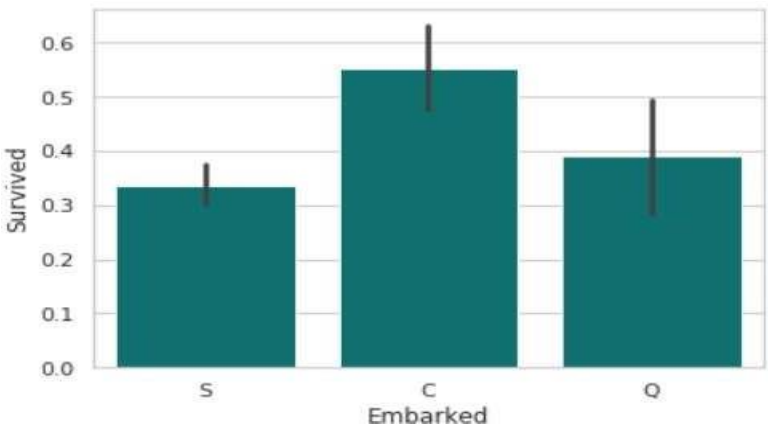
```
plt.figure(figsize=(15,8))

ax=sns.kdeplot(final_train["Age"][final_train.Survived==1],
color="darkturquoise", shade=True)

sns.kdeplot(final_train["Age"][final_train.Survived==0],color="lightcoral",
shade=True)

plt.legend(['Survived', 'Died'])

plt.title('Density Plot of Age for Surviving Population and Deceased Population')

ax.set(xlabel='Age')

plt.xlim(-10,85)

plt.show()

plt.figure(figsize=(15,8))

ax=sns.kdeplot(final_train["Fare"][final_train.Survived==1],
color="darkturquoise", shade=True)

sns.kdeplot(final_train["Fare"][final_train.Survived==0],color="lightcoral",
shade=True)
```




```
plt.figure(figsize=(15,8))

ax=sns.kdeplot(final_train["Fare"][final_train.Survived==1],
color="darkturquoise", shade=True)

sns.kdeplot(final_train["Fare"][final_train.Survived==0],color="lightcoral",
shade=True)

plt.legend(['Survived', 'Died'])

plt.title('Density Plot of Fare for Surviving Population and Deceased Population')

ax.set(xlabel='Fare')

plt.xlim(-20,200)

sns.barplot('Pclass', 'Survived', data=train_df, color="darkturquoise")

plt.show()
```


Density Plot of Fare for Surviving Population and Deceased Population
Survived
Died
0.05
0.04
0.03
0.02
0.01
0.00
0
25
50
75
100
125
150
175
200

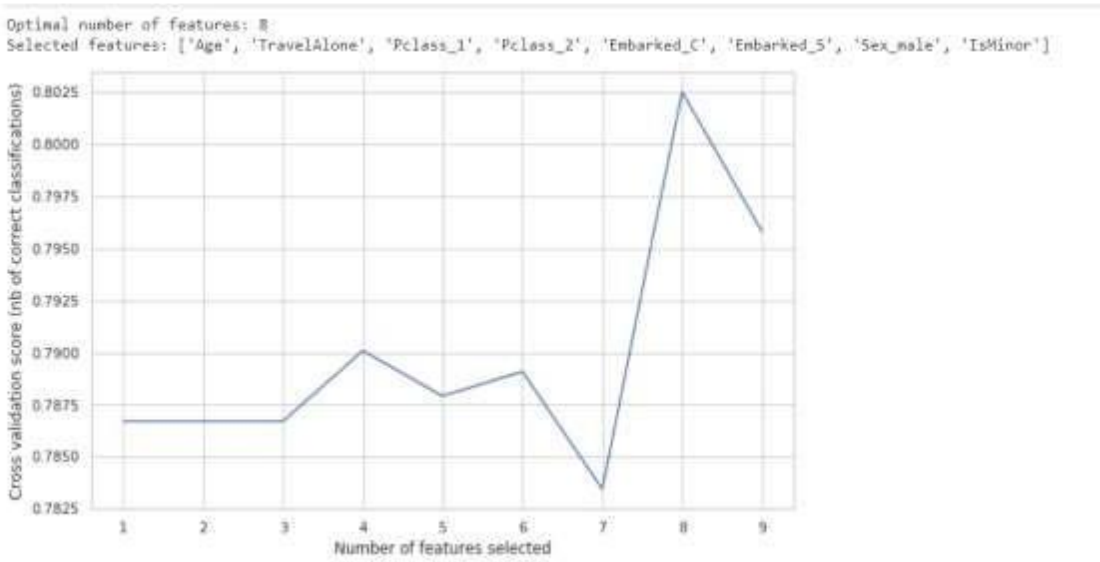

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Survived
1
2
3
Pclass


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Survived
S
C
Q
Embarked

```
from sklearn.linear_model import LogisticRegression

from sklearn.feature_selection import RFE

cols="Age","Fare","TravelAlone","Pclass_1","Pclass_2","Embarked_C","Emba
rked_S","Sex_male","IsMinor"]

X = final_train[cols]

y = final_train['Survived']

model = LogisticRegression()

rfe = RFE(model, 8)

rfe = rfe.fit(X, y)

print('Selected features: %s' % list(X.columns[rfe.support_]))

Selected features: ['Age', 'TravelAlone', 'Pclass_1', 'Pclass_2', 'Embarked_C',
'Embarked_S', 'Sex_male', 'IsMinor']

rfecv=RFECV(estimator=LogisticRegression(),step=1,cv=10,scoring=accuracy)

rfecv.fit(X, y)

print("Optimal number of features: %d" % rfecv.n_features_)

print('Selected features: %s' % list(X.columns[rfecv.support_]))

plt.figure(figsize=(10,6))

plt.xlabel("Number of features selected")

plt.ylabel("Cross validation score (nb of correct classifications)")
```

plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)

plt.show()



Selected_features = ['Age', 'TravelAlone', 'Pclass_1', 'Pclass_2', 'Embarked_C',

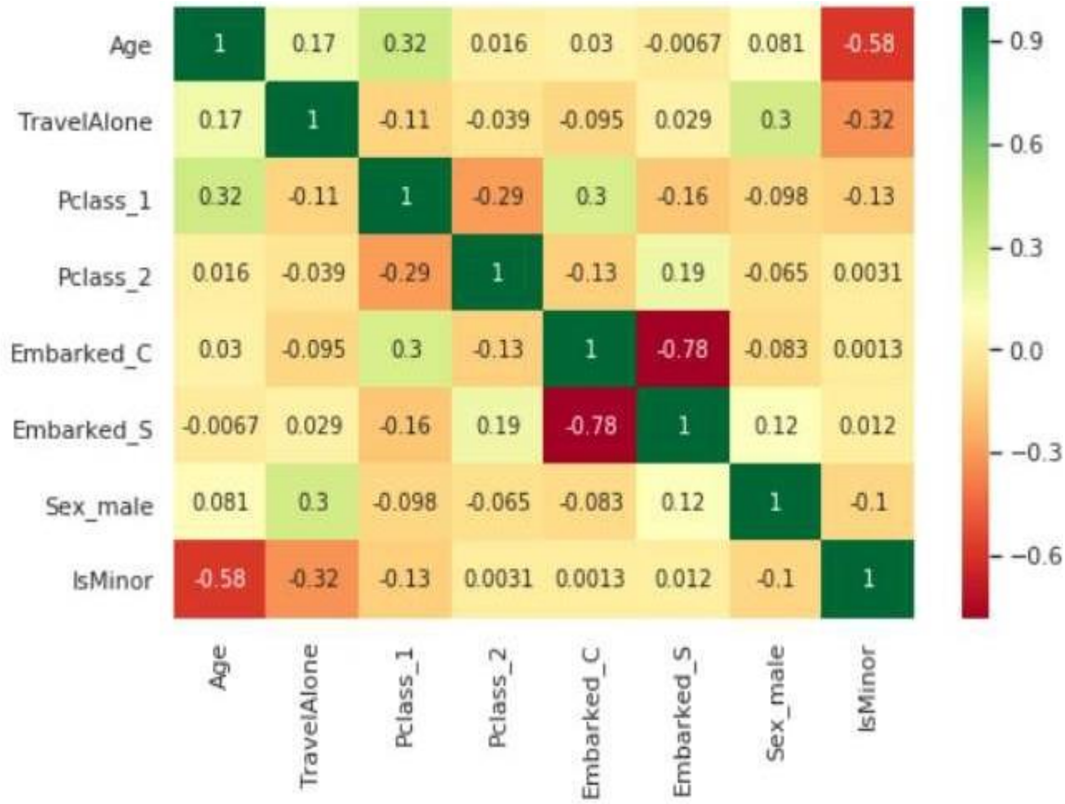
'Embarked_S', 'Sex_male', 'IsMinor']

X = final_train[Selected_features]

plt.subplots(figsize=(8, 5))

sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")

plt.show()



from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.metrics import accuracy_score, classification_report, precision_score, recall_score

from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc, log_loss

X = final_train[Selected_features]

y = final_train['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

```
logreg = LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

y_pred_proba = logreg.predict_proba(X_test)[:, 1]

[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)

print('Train/Test split results:')

print(logreg.__class__.__name__+"accuracy is %2.3f" % accuracy_score(y_test,
y_pred))

print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test,
y_pred_proba))

print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))

idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which the
sensibility > 0.95

plt.figure()

plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % auc(fpr, tpr))

plt.plot([0, 1], [0, 1], 'k--')

plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--', color='blue')

plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--', color='blue')

plt.xlim([0.0, 1.0])
```

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate (1 - specificity)', fontsize=14)

plt.ylabel('True Positive Rate (recall)', fontsize=14)

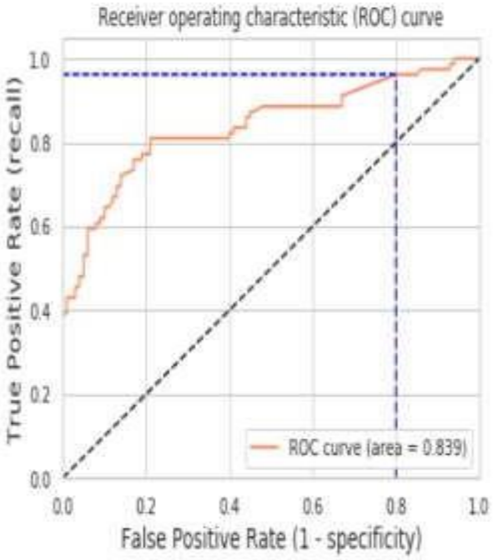
plt.title('Receiver operating characteristic (ROC) curve')

plt.legend(loc="lower right")

plt.show()

print("Using a threshold of %.3f " % thr[idx] + "guarantees a sensitivity of %.3f " % tpr[idx] + "and a specificity of %.3f" % (1-fpr[idx]) +", i.e. a false positive rate of %.2f%%." % (np.array(fpr[idx])*100))

```
Train/Test split results:
LogisticRegression accuracy is 0.782
LogisticRegression log_loss is 0.504
LogisticRegression auc is 0.839
```




```
Using a threshold of 0.071 guarantees a sensitivity of 0.962 and a specificity of 0.200, i.e. a false positive rate of 80.00%.
```

Conclusion:

1. **Features have been chosen to develop the model:**
    1. P_Class: A proxy for socio-economic status (SES)
       1st = Upper, 2nd = Middle, 3rd = Lower
    2. Age: Age is fractional if less than 1.
       If the age is estimated, is it in the form of xx.5
    3. Survival: Yes = 1, No = 0
    4. Sex: Male/Female
    5. Parch: Number of Parents / Childrens
    6. Embarked: Port of Embarkation
       C = Cherbourg, Q = Queenstown, S = Southampton

2. **Accuracy obtained:**
    - We train a logistic regression model on the training set using the Logistic_Regression class from scikit-learn.
    - We then predict the classes of the testing set using the predict method of the model. Finally, we calculate the accuracy of the model using the accuracy_score function from scikit-learn.
    - **LogisticRegression accuracy = 0.782**