In [ ]:

```
#*****************************************************************************************
#
#                 Assignment : Predicting Survival in Titanic Data Set Using Decision Tree
#
#*****************************************************************************************
# Problem statement : Predict the Passenger survived /not , for columns
# Pclass, Sex, Age, SibSp (Siblings aboard), Parch (Parents/children aboard), and Fare

# Step 0 : importing the libraries
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import sklearn
from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import tree,metrics,preprocessing
from sklearn.model_selection import KFold,cross_val_score
from sklearn.metrics import classification_report

Url= "https://raw.githubusercontent.com/BigDataGal/Python-for-DataScience/master/titanic-train.csv"

# Step 1 : Loading Titanic Data
titanic = pd.read_csv("titanic-train.csv")

print(" The Titanic Data for Passengers to predict Survival \n",'-'*80)
titanic.head()

```

In [ ]:

```
#Step 2:  Analysing the Data

print(" Analysing the Data by seeing functions like shape, info , describe, checking for null values \n",'-'*80)
print(titanic.columns)


# number of rows and features
print('-'*80)
print(titanic.shape)
# there are 891 rows and 12 columns

print('-'*80)
print(titanic.info())

print('-'*80)
print(titanic.describe())

print('-'*80)
print(titanic.isnull().sum())


```

In [ ]:
```
#Step 3: Preprocessing the Data

# Since Sex column is categorical , Factrozie Sex Column to numeric value

print(titanic.Sex.unique())
titanic['Sex_num'], _ = pd.factorize(titanic['Sex'])

print("\n Data Set after changing the Sex Field to numeric in Sex_num column ")
titanic.head()

# we observe that Age and Cabin fields have nulls , as the features to be focussed doesnt contain cabin , we can leav
# Filling the null value with mean age for passengers

titanic =  titanic.fillna({'Age':titanic.Age.mean()})

print('-'*80)
print(titanic.isnull().sum())

print(titanic.Age.mean())
```

In [ ]:
```
#Step 4: Visualizing the Data using pairplot to understand the relationships

#print(titanic.head())

predictors = ['Pclass','Sex_num','Age','SibSp','Parch','Fare']
target = 'Survived'

sb.pairplot(titanic,x_vars=predictors,y_vars='Survived',kind="reg")
```

In [ ]:

```
#Step 5: Splitting the Data to train and Test Set with 30 % test data

X= titanic[predictors]
y = titanic[target]

print(" The number of rows and columns before splitting the Data \n",'-'*80)
print(X.shape)
print(y.shape)

y = y.values.reshape(-1,1)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print(" The training set and test set rows and columns after splitting the Data \n",'-'*80)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)


```

In [ ]:

```
# Step 6 : Modeling the Data by using DEcision Tree Classifier and Evaluation by Sklearn metrics

# measure Entropy for the Tree
dtree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=1)

# # train the tree with training Data Set
dtree.fit(X_train,y_train)

# # predicting the results for the Test data set
y_pred = dtree.predict(X_test)
#print(y_pred)

y_pred = y_pred.reshape(-1,1)

print(y_test.shape)
print(y_pred.shape)

misclassified_count = (y_test != y_pred).sum()
print('Misclassified samples: {} / {} and the percentage is {}'.format(misclassified_count,y_test.shape[0],round((mis


accuracy = round(metrics.accuracy_score(y_test, y_pred),2)*100

print('Accuracy of prediction For  Passengers Survival is : {:.2f}  percent'.format(accuracy))

```

In [ ]:

```
# Step 7 : Evaluation by using KFold crossvalidation

from sklearn.model_selection import KFold

print(X.shape)
print(y.shape)


y = pd.DataFrame(y)


dtree1 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)


kf = KFold(n_splits=10, random_state=12)

print(kf)
print(kf.get_n_splits(X))

fold_accuracy = []


for train_fold, valid_fold in kf.split(X):
    train = X.loc[train_fold] # Extract train data with cv indices
    valid = X.loc[valid_fold] # Extract valid data with cv indices

    train_y = y.loc[train_fold]
    valid_y = y.loc[valid_fold]


    train = train.dropna()
    valid = valid.dropna()

    train_y = train_y.dropna()
    valid_y = valid_y.dropna()

    model = dtree1.fit(train,train_y)
    valid_acc = model.score(X = valid, y = valid_y)
    fold_accuracy.append(valid_acc)

print("Accuracy per fold: ", fold_accuracy, "\n")
```

```
42
43 acc = round(sum(fold_accuracy)/len(fold_accuracy),2)*100
44 print("Average accuracy by using KFold is : ", acc)
45
46
```

In [ ]:

```
1  # Model Evaluation by using cross_val_score function
2  from sklearn.model_selection import cross_val_score
3
4  print(X.shape)
5  print(y.shape)
6
7  scores = cross_val_score(dtree1, X, y, scoring = "accuracy", cv=15)
8  print("Accuracy per fold: ")
9  print(scores)
10 print("Average accuracy using the cross_val_Score function : ", round(scores.mean(),2))
```

In [ ]:

```
1
```