**INTRODUCTION :**

We are using Random forest and logistic regression technique. For this dataset logistic regression work better than random forest.

**DATASET :**

We are using the Titanic dataset. train2 contains training data.

The data has been split into two groups:

training set (train.csv)

test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the “ground truth”) for each passenger. Your model will be based on “features” like passengers’ gender and class. You can also use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

We also include gender\_submission.csv, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

Data Dictionary

VariableDefinitionKey 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.

**NOTE :**

**Just upload class2.ipynb in the google colab and just upload the dataset train2 and test2.after that your code is ready to run.**

**CODE :**

***# -\*- coding: utf-8 -\*-***

***"""class2\_titanic.ipynb***

***Automatically generated by Colaboratory.***

***Original file is located at***

***https://colab.research.google.com/drive/1Kniw4k5bCM2pxkoKTcovYvhxzHzDr6g7***

***"""***

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**train=pd.read\_csv('train2.csv')**

**test=pd.read\_csv('test2.csv')**

**train.head()**

**sns.heatmap(train.isnull(),yticklabels=False,cbar=False)**

**train['Age']=train['Age'].fillna(train['Age'].mode()[0])**

**sns.heatmap(train.isnull(),yticklabels=False,cbar=False)**

**train.drop(labels=['Cabin'],axis=1,inplace=True)**

**nan\_cols = [i for i in train.columns if train[i].isnull().any()]**

**nan\_cols**

**train.head()**

**train['Embarked']=train['Embarked'].fillna(train['Embarked'].mode()[0])**

**sns.heatmap(train.isnull(),yticklabels=False,cbar=False)**

**cat\_features=[i for i in train.columns if train.dtypes[i]=='object']**

**cat\_features**

**train.drop(labels=['Name'],axis=1,inplace=True)**

**cat\_features=[i for i in train.columns if train.dtypes[i]=='object']**

**cat\_features**

**def category\_onehot\_multcols(multcolumns):**

**df\_final=train**

**i=0**

**for fields in multcolumns:**

**print(fields)**

**df1=pd.get\_dummies(train[fields],drop\_first=True)**

**train.drop([fields],axis=1,inplace=True)**

**if i==0:**

**df\_final=df1.copy()**

**else:**

**df\_final=pd.concat([df\_final,df1],axis=1)**

**i=i+1**

**df\_final=pd.concat([train,df\_final],axis=1)**

**return df\_final**

**train1=category\_onehot\_multcols(cat\_features)**

**train1.head()**

**"""Here we are using Random forest technique which is ensamble technique"""**

**from sklearn.ensemble import RandomForestClassifier**

**import sklearn**

**from sklearn.model\_selection import train\_test\_split**

**X = train1.drop('Survived', axis=1)**

**y = train1['Survived']**

***#X = pd.get\_dummies(X)***

***#y = pd.get\_dummies(y)***

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)**

**model = RandomForestClassifier(n\_estimators=100, max\_depth=10, random\_state=1)**

**model.fit(X\_train, y\_train)**

**model.score(X\_test, y\_test)**

**"""Here we are using logistic regression with the help of sklearn library"""**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=5)**

**from sklearn.linear\_model import LogisticRegression**

**model = LogisticRegression(max\_iter = 500000)**

**model.fit(X\_train, y\_train)**

**y\_pred = model.predict(X\_test)**

**accuracy = model.score(X\_test, y\_test)**

**print(accuracy)**

**RESULT :**

**Accuracy of random forest is 78.47%**

**And**

**Accuracy of logistic regression is 82.95%**

**For this dataset logistic regression work better than random forest**