Titanic - Machine Learning from Disaster - Machine Learning with Python project

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).

Downloading the dataset

```
import opendatasets as od
```

```
od.download('https://www.kaggle.com/c/titanic/data')
```

```
Please provide your Kaggle credentials to download this dataset. Learn more:
http://bit.ly/kaggle-creds
Your Kaggle username: swetsheersh
Your Kaggle Key: ········

100%|██████████| 34.1k/34.1k [00:00<00:00, 1.23MB/s]

Downloading titanic.zip to .\titanic


Extracting archive .\titanic/titanic.zip to .\titanic
```

```
import os
```

```
os.listdir('./titanic')
```

```
['gender_submission.csv', 'test.csv', 'train.csv']
```

```
import pandas as pd
```

```
gender=pd.read_csv('./titanic/gender_submission.csv')
```

```
test=pd.read_csv('./titanic/test.csv')
train=pd.read_csv('./titanic/train.csv')
```

Problem Statement

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

```
train
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Finding correlation

```
train.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

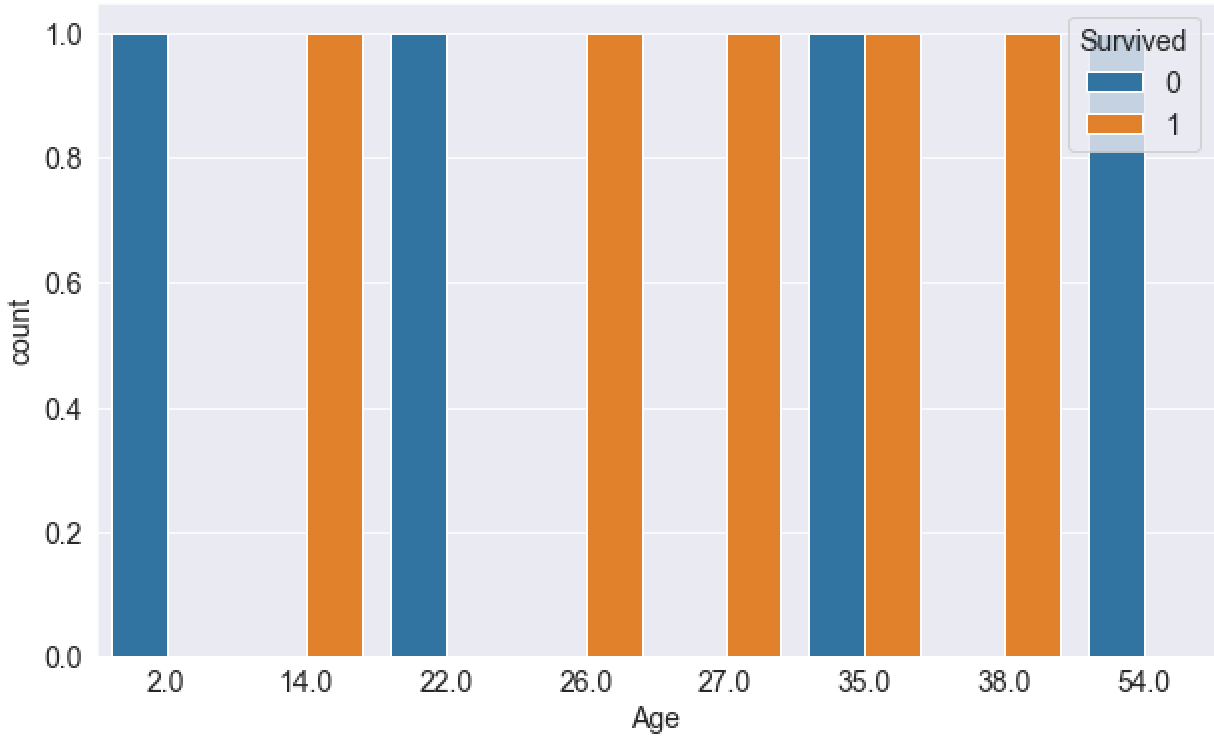
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib
import os
%matplotlib inline

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 150)
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (10, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
import plotly.express as px
```

```
px.scatter(train,x='Age',y='Fare',color='Survived')
```

```python
sns.countplot(x=train['Age'].head(10),hue=train['Survived'])
```

<AxesSubplot:xlabel='Age', ylabel='count'>



Preparing the Data for Training

```
list(train.columns)
```

```
['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked']
```

```
input_cols=[
 'Pclass',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Fare',
 'Cabin',
 'Embarked']
target='Survived'
```

```
x_train=train[input_cols]
train_target=train[target]
x_test=test[input_cols]
```

```
x_train
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	22.0	1	0	7.2500	NaN	S
1	1	female	38.0	1	0	71.2833	C85	C
2	3	female	26.0	0	0	7.9250	NaN	S
3	1	female	35.0	1	0	53.1000	C123	S
4	3	male	35.0	0	0	8.0500	NaN	S
...	...	...	...	...	...	...	...	...
886	2	male	27.0	0	0	13.0000	NaN	S
887	1	female	19.0	0	0	30.0000	B42	S
888	3	female	NaN	1	2	23.4500	NaN	S
889	1	male	26.0	0	0	30.0000	C148	C

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
890	3	male	32.0	0	0	7.7500	NaN	Q

891 rows × 8 columns

```
train_target
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

```
numeric_cols=['Age','SibSp','Parch','Fare']
```

```
cat_cols=['Pclass','Sex','Cabin','Embarked']
```

```
x_train[numeric_cols].isna().sum()
```

```
Age      177
SibSp      0
Parch      0
Fare       0
dtype: int64
```

```
x_test.isna().sum()
```

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

Imputing missing numeric values

```
from sklearn.impute import SimpleImputer
```

```
imputer=SimpleImputer(strategy='mean')
```

```
imputer.fit(x_train[numeric_cols])
```

```
SimpleImputer()
```

```
x_train[numeric_cols]=imputer.transform(x_train[numeric_cols])
x_test[numeric_cols]=imputer.transform(x_test[numeric_cols])
```

```
C:\Users\harsh\anaconda3\lib\site-packages\pandas\core\frame.py:3678:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\harsh\anaconda3\lib\site-packages\pandas\core\frame.py:3678:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Encoding Categorical Data

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder=OneHotEncoder(sparse=False, handle_unknown='ignore').fit(x_train[cat_cols])
```

```
encoded_cols=list(encoder.get_feature_names(cat_cols))
```

```
C:\Users\harsh\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and
```

will be removed in 1.2. Please use get_feature_names_out instead.

```
encoded_cols
```

```
['Pclass_1',
 'Pclass_2',
 'Pclass_3',
 'Sex_female',
 'Sex_male',
 'Cabin_A10',
 'Cabin_A14',
 'Cabin_A16',
 'Cabin_A19',
 'Cabin_A20',
 'Cabin_A23',
 'Cabin_A24',
 'Cabin_A26',
 'Cabin_A31',
 'Cabin_A32',
 'Cabin_A34',
 'Cabin_A36',
 'Cabin_A5',
 'Cabin_A6',
 'Cabin_A7',
 'Cabin_B101',
 'Cabin_B102',
 'Cabin_B18',
 'Cabin_B19',
 'Cabin_B20',
 'Cabin_B22',
 'Cabin_B28',
 'Cabin_B3',
 'Cabin_B30',
 'Cabin_B35',
 'Cabin_B37',
 'Cabin_B38',
 'Cabin_B39',
 'Cabin_B4',
 'Cabin_B41',
 'Cabin_B42',
 'Cabin_B49',
 'Cabin_B5',
 'Cabin_B50',
 'Cabin_B51 B53 B55',
 'Cabin_B57 B59 B63 B66',
 'Cabin_B58 B60',
 'Cabin_B69',
 'Cabin_B71',
```

```
'Cabin_B73',
'Cabin_B77',
'Cabin_B78',
'Cabin_B79',
'Cabin_B80',
'Cabin_B82 B84',
'Cabin_B86',
'Cabin_B94',
'Cabin_B96 B98',
'Cabin_C101',
'Cabin_C103',
'Cabin_C104',
'Cabin_C106',
'Cabin_C110',
'Cabin_C111',
'Cabin_C118',
'Cabin_C123',
'Cabin_C124',
'Cabin_C125',
'Cabin_C126',
'Cabin_C128',
'Cabin_C148',
'Cabin_C2',
'Cabin_C22 C26',
'Cabin_C23 C25 C27',
'Cabin_C30',
'Cabin_C32',
'Cabin_C45',
'Cabin_C46',
'Cabin_C47',
'Cabin_C49',
'Cabin_C50',
'Cabin_C52',
'Cabin_C54',
'Cabin_C62 C64',
'Cabin_C65',
'Cabin_C68',
'Cabin_C7',
'Cabin_C70',
'Cabin_C78',
'Cabin_C82',
'Cabin_C83',
'Cabin_C85',
'Cabin_C86',
'Cabin_C87',
'Cabin_C90',
'Cabin_C91',
'Cabin_C92',
'Cabin_C93',
```

```
'Cabin_C95',
'Cabin_C99',
'Cabin_D',
'Cabin_D10 D12',
'Cabin_D11',
'Cabin_D15',
'Cabin_D17',
'Cabin_D19',
'Cabin_D20',
'Cabin_D21',
'Cabin_D26',
'Cabin_D28',
'Cabin_D30',
'Cabin_D33',
'Cabin_D35',
'Cabin_D36',
'Cabin_D37',
'Cabin_D45',
'Cabin_D46',
'Cabin_D47',
'Cabin_D48',
'Cabin_D49',
'Cabin_D50',
'Cabin_D56',
'Cabin_D6',
'Cabin_D7',
'Cabin_D9',
'Cabin_E10',
'Cabin_E101',
'Cabin_E12',
'Cabin_E121',
'Cabin_E17',
'Cabin_E24',
'Cabin_E25',
'Cabin_E31',
'Cabin_E33',
'Cabin_E34',
'Cabin_E36',
'Cabin_E38',
'Cabin_E40',
'Cabin_E44',
'Cabin_E46',
'Cabin_E49',
'Cabin_E50',
'Cabin_E58',
'Cabin_E63',
'Cabin_E67',
'Cabin_E68',
'Cabin_E77',
```

```
 'Cabin_E8',
 'Cabin_F E69',
 'Cabin_F G63',
 'Cabin_F G73',
 'Cabin_F2',
 'Cabin_F33',
 'Cabin_F38',
 'Cabin_F4',
 'Cabin_G6',
 'Cabin_T',
 'Cabin_nan',
 'Embarked_C',
 'Embarked_Q',
 'Embarked_S',
 'Embarked_nan']
```

```
len(encoded_cols)
```

```
157
```

```
x_train[encoded_cols]=encoder.transform(x_train[cat_cols])
x_test[encoded_cols]=encoder.transform(x_test[cat_cols])
```

```
C:\Users\harsh\anaconda3\lib\site-packages\pandas\core\frame.py:3678:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\harsh\anaconda3\lib\site-packages\pandas\core\frame.py:3678:
PerformanceWarning:

DataFrame is highly fragmented.  This is usually the result of calling `frame.insert`
many times, which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead.  To get a de-fragmented frame, use `newframe = frame.copy()`

C:\Users\harsh\anaconda3\lib\site-packages\pandas\core\frame.py:3678:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\harsh\anaconda3\lib\site-packages\pandas\core\frame.py:3678:
PerformanceWarning:

DataFrame is highly fragmented.  This is usually the result of calling `frame.insert`
many times, which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead.  To get a de-fragmented frame, use `newframe = frame.copy()`
```

Scaling Numeric Features

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler=MinMaxScaler().fit(x_train[numeric_cols])
```

```
x_train[numeric_cols].describe().loc[['min', 'max']]
```

	Age	SibSp	Parch	Fare
min	0.42	0.0	0.0	0.0000
max	80.00	8.0	6.0	512.3292

```
x_train[numeric_cols]=scaler.transform(x_train[numeric_cols])
```

```
x_test[numeric_cols]=scaler.transform(x_test[numeric_cols])
```

```
x_train=x_train[numeric_cols + encoded_cols]
```

```
x_test=x_test[numeric_cols+ encoded_cols]
```

```
x_train
```

	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Cabin_A10	Cabin_
0	0.271174	0.125	0.000000	0.014151	0.0	0.0	1.0	0.0	1.0	0.0	
1	0.472229	0.125	0.000000	0.139136	1.0	0.0	0.0	1.0	0.0	0.0	
2	0.321438	0.000	0.000000	0.015469	0.0	0.0	1.0	1.0	0.0	0.0	
3	0.434531	0.125	0.000000	0.103644	1.0	0.0	0.0	1.0	0.0	0.0	
4	0.434531	0.000	0.000000	0.015713	0.0	0.0	1.0	0.0	1.0	0.0	

	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Cabin_A10	Cabin_
...	...	...	...	...	...	...	...	...	...	...	
886	0.334004	0.000	0.000000	0.025374	0.0	1.0	0.0	0.0	1.0	0.0	
887	0.233476	0.000	0.000000	0.058556	1.0	0.0	0.0	1.0	0.0	0.0	
888	0.367921	0.125	0.333333	0.045771	0.0	0.0	1.0	1.0	0.0	0.0	
889	0.321438	0.000	0.000000	0.058556	1.0	0.0	0.0	0.0	1.0	0.0	
890	0.396833	0.000	0.000000	0.015127	0.0	0.0	1.0	0.0	1.0	0.0	

891 rows × 161 columns

```
x_test
```

	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Cabin_A10	Cabin_
0	0.428248	0.000	0.000000	0.015282	0.0	0.0	1.0	0.0	1.0	0.0	
1	0.585323	0.125	0.000000	0.013663	0.0	0.0	1.0	1.0	0.0	0.0	
2	0.773813	0.000	0.000000	0.018909	0.0	1.0	0.0	0.0	1.0	0.0	
3	0.334004	0.000	0.000000	0.016908	0.0	0.0	1.0	0.0	1.0	0.0	
4	0.271174	0.125	0.166667	0.023984	0.0	0.0	1.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
413	0.367921	0.000	0.000000	0.015713	0.0	0.0	1.0	0.0	1.0	0.0	
414	0.484795	0.000	0.000000	0.212559	1.0	0.0	0.0	1.0	0.0	0.0	
415	0.478512	0.000	0.000000	0.014151	0.0	0.0	1.0	0.0	1.0	0.0	
416	0.367921	0.000	0.000000	0.015713	0.0	0.0	1.0	0.0	1.0	0.0	
417	0.367921	0.125	0.166667	0.043640	0.0	0.0	1.0	0.0	1.0	0.0	

418 rows × 161 columns

Training, Validation and Test Sets

```
from sklearn.model_selection import train_test_split
```

```
train_df,val_df,target,val_target = train_test_split(x_train,train_target, test_size=0.
```

```
train_df
```

	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Cabin_A10	Cabin_
331	0.566474	0.000	0.000000	0.055628	1.0	0.0	0.0	0.0	1.0	0.0	
733	0.283740	0.000	0.000000	0.025374	0.0	1.0	0.0	0.0	1.0	0.0	
382	0.396833	0.000	0.000000	0.015469	0.0	0.0	1.0	0.0	1.0	0.0	

	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Cabin_A10	Cabin_
704	0.321438	0.125	0.000000	0.015330	0.0	0.0	1.0	0.0	1.0	0.0	
813	0.070118	0.500	0.333333	0.061045	0.0	0.0	1.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
106	0.258608	0.000	0.000000	0.014932	0.0	0.0	1.0	1.0	0.0	0.0	
270	0.367921	0.000	0.000000	0.060508	1.0	0.0	0.0	0.0	1.0	0.0	
860	0.509927	0.250	0.000000	0.027538	0.0	0.0	1.0	0.0	1.0	0.0	
435	0.170646	0.125	0.333333	0.234224	1.0	0.0	0.0	1.0	0.0	0.0	
102	0.258608	0.000	0.166667	0.150855	1.0	0.0	0.0	0.0	1.0	0.0	

712 rows × 161 columns

```
target
```

```
331    0
733    0
382    0
704    0
813    0
      ..
106    1
270    0
860    0
435    1
102    0
Name: Survived, Length: 712, dtype: int64
```

```
val_df
```

	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Cabin_A10	Cabin_
709	0.367921	0.125	0.166667	0.029758	0.0	0.0	1.0	0.0	1.0	0.0	
439	0.384267	0.000	0.000000	0.020495	0.0	1.0	0.0	0.0	1.0	0.0	
840	0.246042	0.000	0.000000	0.015469	0.0	0.0	1.0	0.0	1.0	0.0	
720	0.070118	0.000	0.166667	0.064412	0.0	1.0	0.0	1.0	0.0	0.0	
39	0.170646	0.125	0.000000	0.021942	0.0	0.0	1.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
433	0.208344	0.000	0.000000	0.013907	0.0	0.0	1.0	0.0	1.0	0.0	
773	0.367921	0.000	0.000000	0.014102	0.0	0.0	1.0	0.0	1.0	0.0	
25	0.472229	0.125	0.833333	0.061264	0.0	0.0	1.0	1.0	0.0	0.0	
84	0.208344	0.000	0.000000	0.020495	0.0	1.0	0.0	1.0	0.0	0.0	
10	0.044986	0.125	0.166667	0.032596	0.0	0.0	1.0	1.0	0.0	0.0	

179 rows × 161 columns

```
val_target
```

```
709    1
439    0
840    0
720    1
39     1
      ..
433    0
773    0
25     1
84     1
10     1
Name: Survived, Length: 179, dtype: int64
```

Training LogisticRegression

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(solver='liblinear')
```

```
model.fit(train_df,target)
```

```
LogisticRegression(solver='liblinear')
```

```
print(model.coef_.tolist())
```

```
[[-1.4533535881468072, -1.2479475461997147, -0.5650237925947144, 0.6557067917670618,
0.5062863184064099, 0.6906279296016066, -0.46565744591367203, 1.6617683367379457,
-0.9305115346436355, -0.4360541726513401, -0.3442986936165871, 0.0777080878151226,
-0.33819645556695577, 0.0, 0.6459052672182573, -0.3395811950559433, 0.4387768062023739,
0.0, -0.34382776664090653, 0.3973927343486333, -0.3019192156761674, 0.0, 0.0, 0.0,
0.25767060703716904, -0.3321380971824785, 0.10567506958506244, -0.24270391632435886,
0.5360228671437278, -0.08063729771163532, 0.16531029892975635, 0.09500404914268452,
0.0, 0.08515617949023493, -0.356095270377807 9, -0.28889032989026486, 0.0,
0.06449398468383263, 0.49200401194316074, 0.07354535231155279, 0.3463844577462915,
0.12464395666626092, 0.35553046142494027, -0.042473868317895166, 0.05305545759273116,
-0.5138160649136825, 0.0, -0.3118434504800845, 0.0815344117716543, 0.15733752386010458,
0.0, 0.06583312541590892, 0.07065614002399662, -0.4100326032805617, 0.0,
-0.2987479656122266, 0.6319925008969997, 0.15342691195159938, 0.1328209198474658,
0.5468901182853232, 0.4655849514724644, 0.0, -0.45462348068089214,
-0.41137235759915336, 0.1046984627492525, -0.5354271451423506, 0.19887808737541163,
0.0, -0.340303878264133, 0.0, -0.22962075119329736, -0.9086543029317709,
```

```
-0.23347101378766927, -0.25943557651780197, 0.050036851377682705, 0.04649745041416073,
-0.30184906693071584, 0.34791798366325855, -0.8401812344147981, 0.07965090262696857,
0.773101819664713, 0.043553638806235626, 0.03836103307961725, -0.4334450865858846,
-0.2836813157358636, 0.0899296891737716, 0.3210973651283791, -0.23937401966420244,
-0.4748256632820243, -0.1679314556941081, 0.06397385905818498, -0.3726818833297934,
-0.23851039391035495, 0.0452998926990381, -0.3231904215410915, 0.4308094931543983,
0.5645005532321268, 0.0, 0.08418375043033129, 0.03919841112589497, 0.0,
0.12899031243071707, 0.0, 0.11503547312683374, 0.0, 0.08182504055358591, 0.0,
-0.5320193311491663, 0.0, -0.3518441724741055, 0.47237356709176237, 0.6072594121151683,
0.047689984467276514, 0.09008179071663355, 0.4657123659919896, -0.2843113243467802,
0.0, 0.0, 0.32791450495282704, 0.0, 0.4489854540988772, -0.34147338723197884, 0.0,
0.07198688004748917, 0.6685926790898316, 0.18528685611516102, 0.5333604058759979,
0.6984054588848276, 0.5442653679564025, 0.8323601464568984, 0.0, -0.2671642216834539,
0.16156229512343184, 0.0, 0.0, -0.23017654191435324, 0.054405254128372764,
-0.1388399902325481, -0.26683640302434436, 0.0, 0.35483590237450613,
-0.28249130333520317, 0.0, -0.11802212233233983, 0.0, 0.0, 0.5357707145755802,
0.14365230243033358, -0.15355325540519907, -0.353072834766533, 0.3254938908575582,
0.14141225760573073, -0.23352541097176582, 0.44335502206718763, -0.8354813099269137,
-0.2908436865084065, -0.9819113277994159, 0.47007394438493605, 0.23471469829683406,
-0.13884213951718033, 0.16531029892975635]]
```

```python
print(model.intercept_)
```

```
[0.7312568]
```

```python
train_preds = model.predict(train_df)
```

```python
train_probs = model.predict_proba(train_df)
train_probs
```

```
array([[0.76024968, 0.23975032],
       [0.73596797, 0.26403203],
       [0.91311386, 0.08688614],
       ...,
       [0.94378424, 0.05621576],
       [0.04956189, 0.95043811],
       [0.67588667, 0.32411333]])
```

```python
from sklearn.metrics import accuracy_score,confusion_matrix
```

```python
accuracy_score(target, train_preds)
```

```
0.8384831460674157
```

```python
accuracy_score(val_target, model.predict(val_df))
```

```
0.8100558659217877
```

```
confusion_matrix(train_preds, target, normalize='true')
```

```
array([[0.85683297, 0.14316703],
       [0.19521912, 0.80478088]])
```

From Logistic Regression I got Accuracy of 81% on validation data set

DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree=DecisionTreeClassifier(random_state=42)
```

```
tree.fit(train_df, target)
```

```
DecisionTreeClassifier(random_state=42)
```

```
train_preds1 = tree.predict(train_df)
```

```
accuracy_score(target,train_preds1)
```

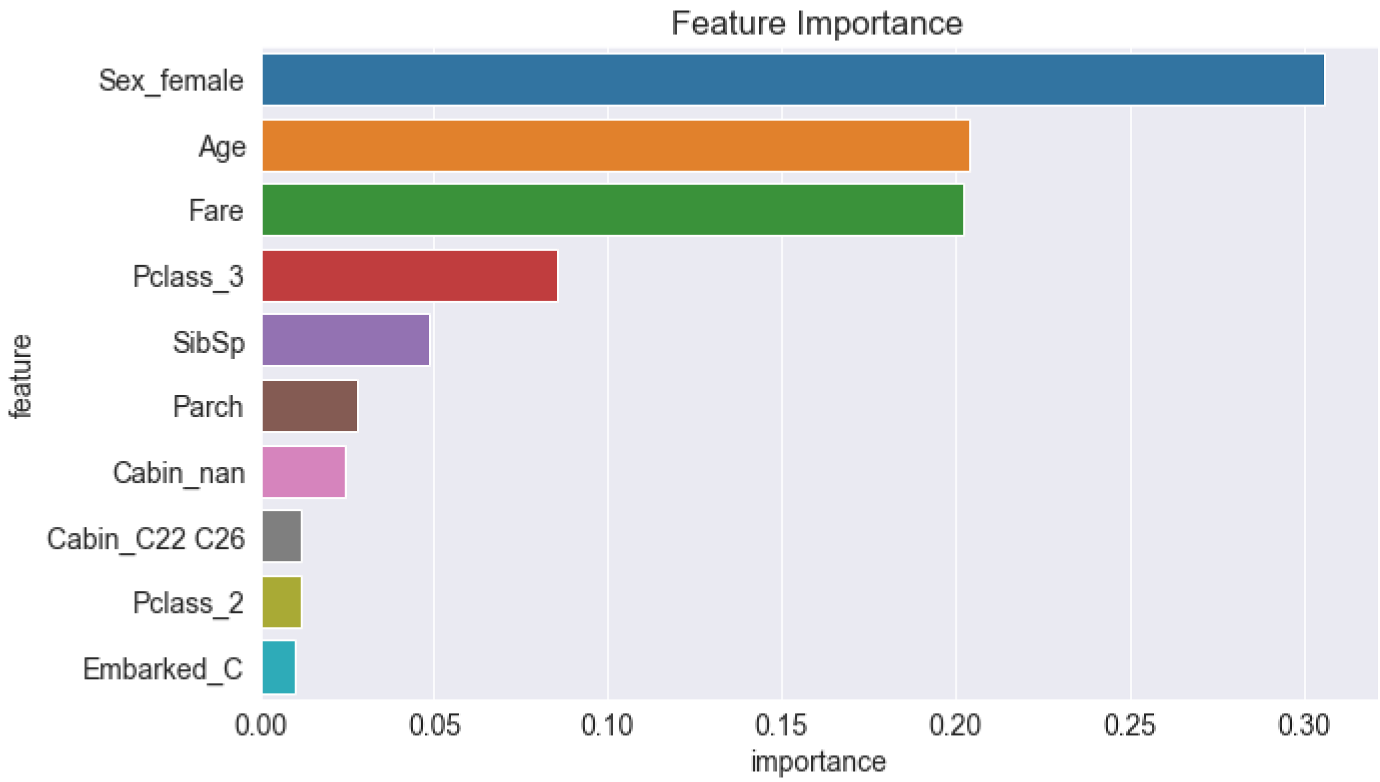
```
0.9845505617977528
```

```
accuracy_score(val_target, tree.predict(val_df))
```

```
0.8044692737430168
```

```
importance_df = pd.DataFrame({
    'feature':train_df.columns,
    'importance': tree.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```




```
model = DecisionTreeClassifier(max_depth=9, random_state=42)
```

```
model.fit(train_df, target)
```

```
DecisionTreeClassifier(max_depth=9, random_state=42)
```

```
model.score(train_df, target)
```

```
0.9185393258426966
```

```
model.score(val_df, val_target)
```

```
0.8324022346368715
```

```
def max_depth_error(md):
    model = DecisionTreeClassifier(max_depth=md, random_state=42)
    model.fit(train_df, target)
    train_acc = 1 - model.score(train_df, target)
    val_acc = 1 - model.score(val_df, val_target)
    return {'Max Depth': md, 'Training Error': train_acc, 'Validation Error': val_acc}
```

```
%%time
errors_df = pd.DataFrame([max_depth_error(md) for md in range(1, 21)])
```

```
Wall time: 217 ms
```

```
errors_df
```

	Max Depth	Training Error	Validation Error

	Max Depth	Training Error	Validation Error
0	1	0.212079	0.217877
1	2	0.196629	0.234637
2	3	0.162921	0.201117
3	4	0.160112	0.201117
4	5	0.150281	0.201117
5	6	0.129213	0.184358
6	7	0.112360	0.178771
7	8	0.095506	0.178771
8	9	0.081461	0.167598
9	10	0.067416	0.195531
10	11	0.049157	0.206704
11	12	0.042135	0.189944
12	13	0.033708	0.206704
13	14	0.028090	0.195531
14	15	0.023876	0.206704
15	16	0.019663	0.201117
16	17	0.015449	0.201117
17	18	0.015449	0.189944
18	19	0.015449	0.201117
19	20	0.015449	0.206704

From DecisionTree I got Accuracy of 83% on validation data set

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(n_jobs=-1, random_state=42)
```

```
model.fit(train_df, target)
```

```
RandomForestClassifier(n_jobs=-1, random_state=42)
```

```
model.score(train_df, target)
```

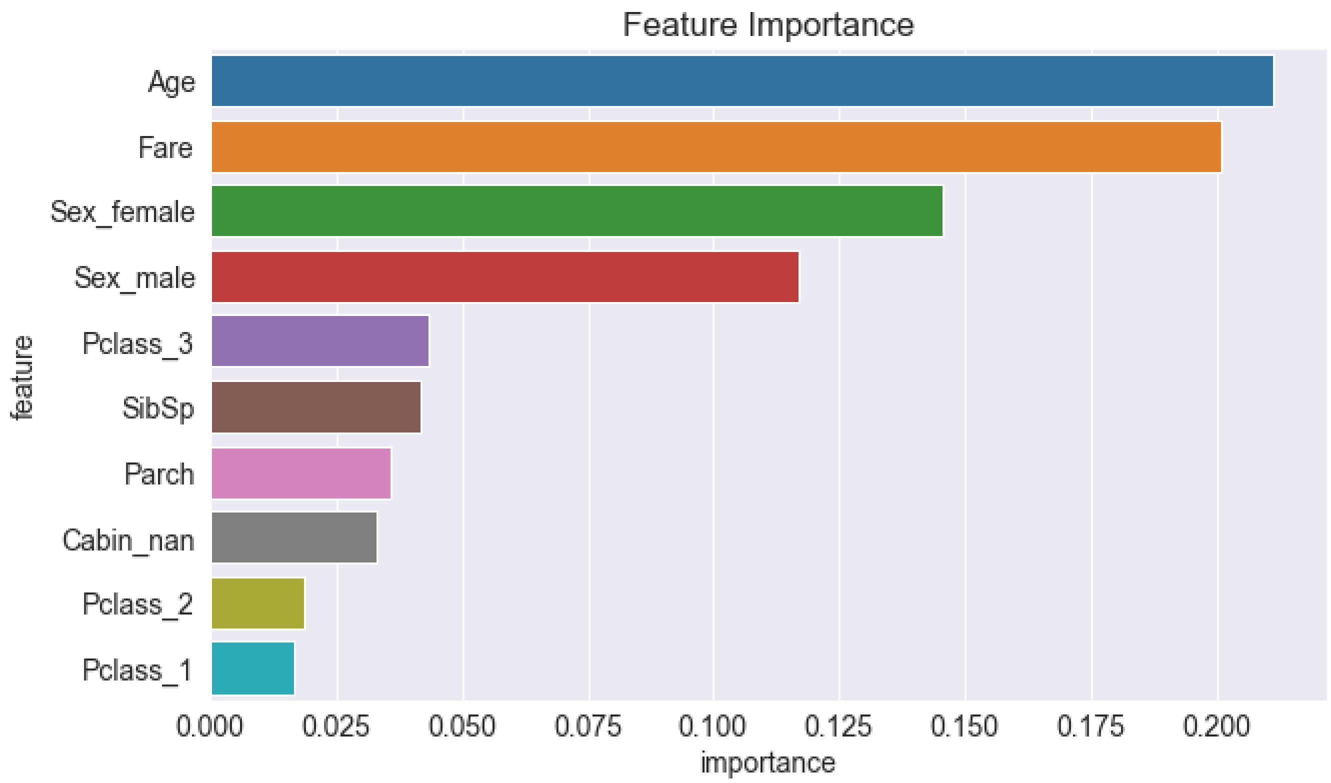
```
0.9845505617977528
```

```
model.score(val_df, val_target)
```

```
0.7988826815642458
```

```
importance_df = pd.DataFrame({
    'feature': train_df.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```




```
model = RandomForestClassifier(random_state=42, n_jobs=-1, n_estimators=500)
```

```
model.fit(train_df, target)
```

```
RandomForestClassifier(n_estimators=500, n_jobs=-1, random_state=42)
```

```
model.score(train_df, target)
```

```
0.9845505617977528
```

```
model.score(val_df, val_target)
```

```
0.7932960893854749
```

```
def test_params(**params):
    model = RandomForestClassifier(random_state=42, n_jobs=-1, **params).fit(train_df,
    return model.score(train_df, target), model.score(val_df, val_target)
```

```
test_params(max_depth=5)
```

```
(0.8356741573033708, 0.8100558659217877)
```

```
test_params(max_depth=26)
```

```
(0.9845505617977528, 0.7988826815642458)
```

```
test_params(max_leaf_nodes=2**5)
```

```
(0.8792134831460674, 0.8156424581005587)
```

```
test_params(max_leaf_nodes=2**20)
```

```
(0.9845505617977528, 0.7988826815642458)
```

```
test_params(max_features='log2')
```

```
(0.9845505617977528, 0.7932960893854749)
```

```
test_params(min_samples_split=100, min_samples_leaf=60)
```

```
(0.7036516853932584, 0.7094972067039106)
```

```
test_params(bootstrap=False)
```

```
(0.9845505617977528, 0.8100558659217877)
```

```
test_params(class_weight='balanced')
```

```
(0.9845505617977528, 0.8100558659217877)
```

From Random Forest I got Accuracy of 79% on validation data set

xgboost

```
from xgboost import XGBClassifier
```

```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=100,max_depth=10,learning_
```

```
model.fit(train_df, target)
```

```
C:\Users\harsh\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
```

```
[15:09:11] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=10, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
model.score(train_df, target)
```

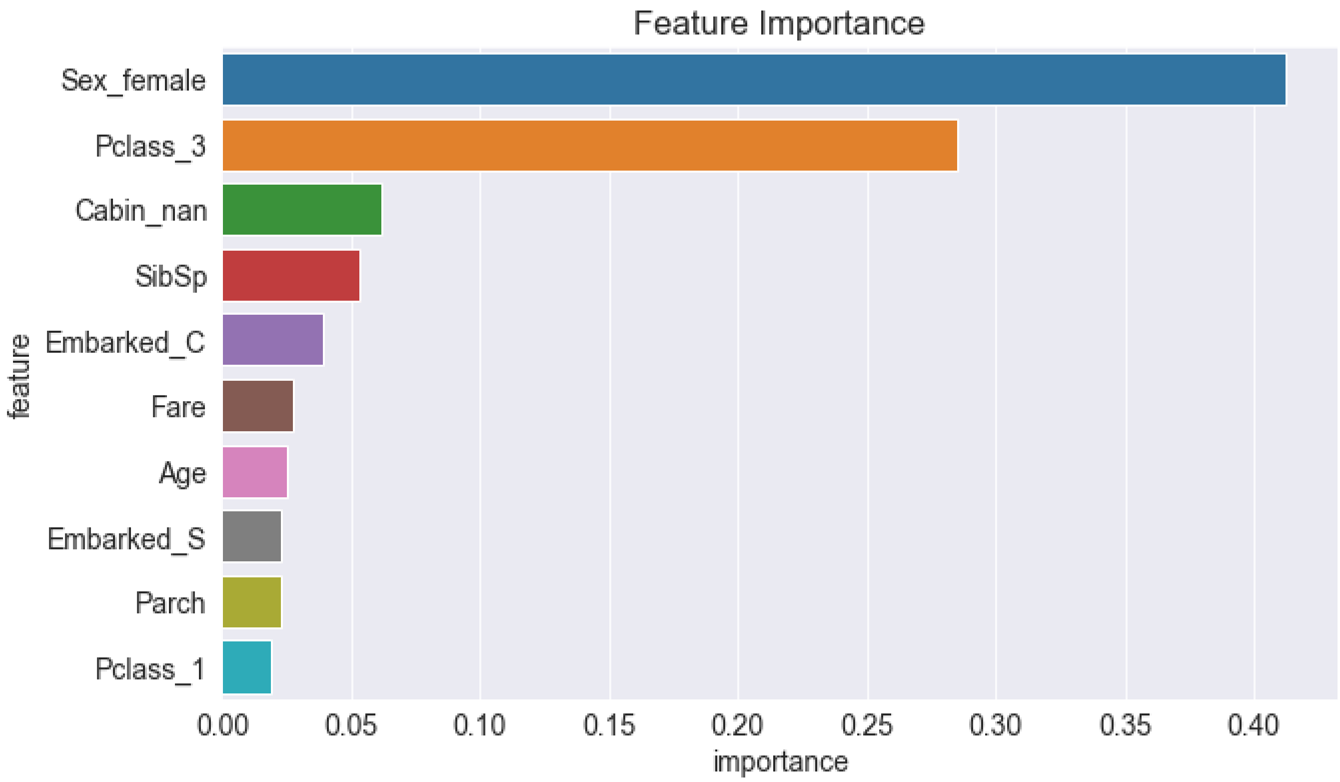
```
0.9817415730337079
```

```
model.score(val_df, val_target)
```

```
0.8212290502793296
```

```
importance_df = pd.DataFrame({
    'feature': train_df.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```




```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=100,max_depth=10,learning_
```

```
test
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

```
gender
```

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

```
model.fit(x_train,train_target)
```

```
C:\Users\harsh\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[15:17:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=10, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
model.score(x_train,train_target)
```

```
0.9854096520763187
```

```
preds=model.predict(x_test)
```

```
gender
```

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	1
4	896	0
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

From XGBOOST I got Accuracy of 82% on validation data set

```
gender['Survived']=preds
```

```
gender=gender[['PassengerId','Survived']]
```

Saving Result

```
gender.to_csv('submission.csv', index=None)
```

```
from IPython.display import FileLink
```

```
# Doesn't work on Colab, use the file browser instead.
FileLink('submission.csv')
```

[submission.csv]

```
model = DecisionTreeClassifier(max_depth=9, random_state=42)
```

```
model.fit(x_train,train_target)
```

```
DecisionTreeClassifier(max_depth=9, random_state=42)
```

```
model.score(x_train,train_target)
```

```
0.9169472502805837
```

```
preds=model.predict(x_test)
```

```
gender['Survived']=preds
```

```
<ipython-input-515-f9aa3be22856>:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
gender
```

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	1

418 rows × 2 columns

```
gender.to_csv('submission1.csv', index=None)
```

Saving and Loading Trained Models

```
import joblib
```

```
titanic = {
    'model': model,
    'imputer': imputer,
    'scaler': scaler,
    'encoder': encoder,
    'input_cols': input_cols,
```

```
    'target_col': target_col,
    'numeric_cols': numeric_cols,
    'categorical_cols': cat_cols,
    'encoded_cols': encoded_cols
}
```

```
joblib.dump(titanic, 'titanic.joblib')
```

```
['titanic.joblib']
```

```
titanic2 = joblib.load('titanic.joblib')
```

Summary and References

Finally I got Accuracy of 83% on Validation set.

YOU can download the Dataset from Kaggle:

- https://www.kaggle.com/c/titanic/overview

```

```