Titanic - Machine Learning from Disaster

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder,OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import seaborn as sns
import matplotlib.pyplot as plt
```

In [163]:
```
from pathlib import Path
import pandas as pd
import tarfile
import urllib.request

def load_titanic_data():
    tarball_path = Path("datasets/titanic.tgz")
    if not tarball_path.is_file():
        Path("datasets").mkdir(parents=True, exist_ok=True)
        url = "https://github.com/ageron/data/raw/main/titanic.tgz"
        urllib.request.urlretrieve(url, tarball_path)
        with tarfile.open(tarball_path) as titanic_tarball:
            titanic_tarball.extractall(path="datasets")
    return [pd.read_csv(Path("datasets/titanic") / filename)
            for filename in ("train.csv", "test.csv")]
```

In [164]:
```
train_data, test_data = load_titanic_data()
```

In [165]:
```
train_data.head()
```

Out[165]:

	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

In [166]:
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [167]:
```
train_data['Cabin'].unique()
```

Out[167]:
```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

In [168]:
```
train_data = train_data.drop(['Cabin','Name'],axis = 1)
```

In [169]:
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Sex          891 non-null    object 
 4   Age          714 non-null    float64
 5   SibSp        891 non-null    int64  
 6   Parch        891 non-null    int64  
 7   Ticket       891 non-null    object 
 8   Fare         891 non-null    float64
 9   Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(3)
memory usage: 69.7+ KB
```

In [170]:
```
train_data['Survived'].value_counts()
```

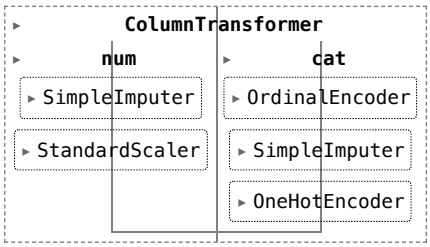
Out[170]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

In [215]:
```
gender_stats = train_data.groupby('Sex').agg({'Sex':'count','Survived':'sum'})
gender_stats = gender_stats.rename(columns={'Sex':'Total'})


print(f"Percentage of males survived is {round(gender_stats.loc['male']['Survived']/gender_stats.loc['male']
print(f"Percentage of females survived is {round(gender_stats.loc['female']['Survived']/gender_stats.loc['f
```

```
Percentage of males survived is 18.89
Percentage of females survived is 74.2
```

In [171]:
```
num_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy = 'median')),
    ('scaler',StandardScaler())
])

cat_pipeline = Pipeline([
    ('ordinal_encoder',OrdinalEncoder()),
    ('imputer',SimpleImputer(strategy = 'most_frequent')),
    ("cat_encoder", OneHotEncoder(sparse_output=False))
])

num_attribs = ["Age", "SibSp", "Parch", "Fare"]
cat_attribs = ["Pclass", "Sex", "Embarked"]


preprocessing = ColumnTransformer([
    ('num',num_pipeline,num_attribs),
    ('cat',cat_pipeline,cat_attribs),
])

preprocessing
```

Out[171]:



In [172]:
```
train_data.head()
```

Out[172]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	female	35.0	1	0	113803	53.1000	S
4	5	0	3	male	35.0	0	0	373450	8.0500	S

In [173]:
```
X_train = preprocessing.fit_transform(train_data)
X_train
```

Out[173]:
```
array([[-0.56573582,  0.43279337, -0.47367361, ...,  0.        ,
         0.        ,  1.        ],
       [ 0.6638609 ,  0.43279337, -0.47367361, ...,  1.        ,
         0.        ,  0.        ],
       [-0.25833664, -0.4745452 , -0.47367361, ...,  0.        ,
         0.        ,  1.        ],
       ...,
       [-0.10463705,  0.43279337,  2.00893337, ...,  0.        ,
         0.        ,  1.        ],
       [-0.25833664, -0.4745452 , -0.47367361, ...,  1.        ,
         0.        ,  0.        ],
       [ 0.20276213, -0.4745452 , -0.47367361, ...,  0.        ,
         1.        ,  0.        ]])
```

In [174]:
```
y_train = train_data['Survived']
y_train
```

Out[174]:
```
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
```

In [175]:
```
X_test = preprocessing.transform(test_data)
```

In [176]:
```
rnd_clf = RandomForestClassifier(random_state = 449)

rnd_clf.fit(X_train,y_train)
```

Out[176]:
```
RandomForestClassifier(random_state=449)
```

In [177]:
```
y_pred = rnd_clf.predict(X_test)
```

In [178]:
```
forest_score = cross_val_score(rnd_clf,X_train,y_train,cv=3,scoring = 'accuracy')

forest_score
```

Out[178]:
```
array([0.79124579, 0.81481481, 0.79124579])
```

In [179]:
```
sgd_clf = SGDClassifier(random_state = 449)

sgd_clf.fit(X_train,y_train)

sgd_score = cross_val_score(sgd_clf,X_train,y_train,cv=3,scoring = 'accuracy')

print("The accuracy of SGD is ",sgd_score.max())
```

```
The accuracy of SGD is  0.7946127946127947
```

In [180]:
```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
    'gamma': ['scale', 'auto']
}

grid_search = GridSearchCV(SVC(),param_grid,scoring = 'accuracy')

grid_search.fit(X_train,y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)
```

```
{'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}
0.828278199736363
```

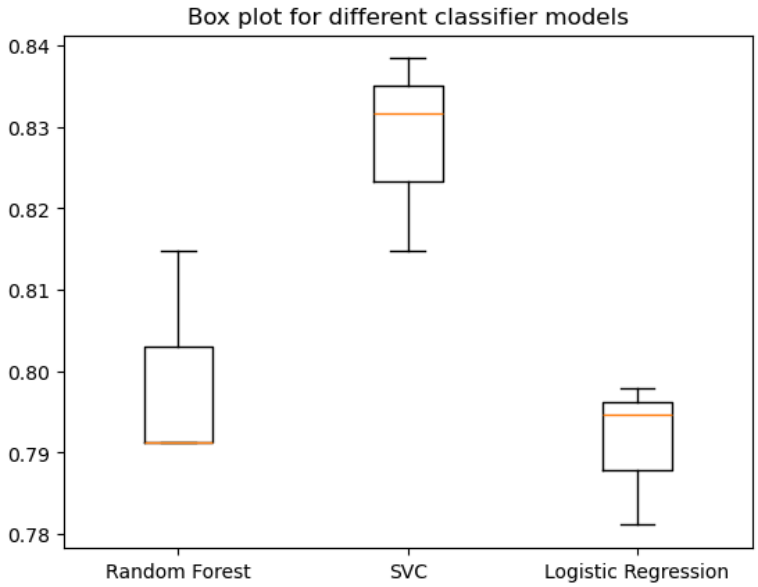
In [181]:
```
svc_clf = SVC(random_state = 449)

svc_clf.fit(X_train,y_train)

svc_score = cross_val_score(svc_clf,X_train,y_train,cv=3,scoring = 'accuracy')

print("The accuracy of SVC is ",svc_score.max())
```

```
The accuracy of SVC is  0.8383838383838383
```

In [182]:
```
log_reg = LogisticRegression()

log_reg.fit(X_train,y_train)

log_reg_score = cross_val_score(log_reg,X_train,y_train,cv=3,scoring = 'accuracy')

print("The accuracy of LogisticRegression is ",log_reg_score.max())
```

```
The accuracy of LogisticRegression is  0.797979797979798
```

In [188]:
```
plt.boxplot([forest_score,svc_score,log_reg_score],labels = ['Random Forest','SVC','Logistic Regression'])
plt.title('Box plot for different classifier models')
plt.show()
```



-> SVC has better accuracy compared to Random Forest and Logistic Regression.

In [186]:
```
submission = pd.DataFrame({'PassengerId':test_data.PassengerId,'Survived':y_pred})

submission = submission.reset_index(drop = True)

submission.to_csv('Submission.csv',index = False)
```

In [187]:
```
print("The accuracy of SVC is ",svc_score.max())
```

```
The accuracy of SVC is  0.8383838383838383
```