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

plt.style.use('seaborn')
sns.set(font_scale=2.5)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

#ignore warnings
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

# 1. Preprocessing

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

df_train['FamilySize'] = df_train['SibSp'] + df_train['Parch'] + 1 # 1 for the passenger himself
df_test['FamilySize'] = df_test['SibSp'] + df_test['Parch'] + 1 # 1 for the passenger himself

df_test.loc[df_test.Fare.isnull(), 'Fare'] = df_test['Fare'].mean()

df_train['Fare'] = df_train['Fare'].map(lambda i: np.log(i) if i > 0 else 0)
df_test['Fare'] = df_test['Fare'].map(lambda i: np.log(i) if i > 0 else 0)
```

## 1.1 Fill Null

```
df_train['Initial']= df_train.Name.str.extract('([A-Za-z]+)\.') #lets extract the Salutations
df_test['Initial']= df_test.Name.str.extract('([A-Za-z]+)\.') #lets extract the Salutations
```

```
pd.crosstab(df_train['Initial'], df_train['Sex']).T.style.background_gradient(cmap='summer_r') #Checking the Initials with the Sex
```

| Initial | Capt | Col | Countess | Don | Dr | Jonkheer | Lady | Major | Master | Miss | Mlle | Mme | Mr | Mrs | Ms | Rev | Sir |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sex | | | | | | | | | | | | | | | | | |
| female | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 182 | 2 | 1 | 0 | 125 | 1 | 0 | 0 |
| male | 1 | 2 | 0 | 1 | 6 | 1 | 0 | 2 | 40 | 0 | 0 | 0 | 517 | 0 | 0 | 6 | 1 |

```
df_train['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don', 'Dona'],
                            ['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr', 'Mr'],inplace=True)

df_test['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don', 'Dona'],
                            ['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr', 'Mr'],inplace=True)
```

```
df_train.groupby('Initial').mean()
```

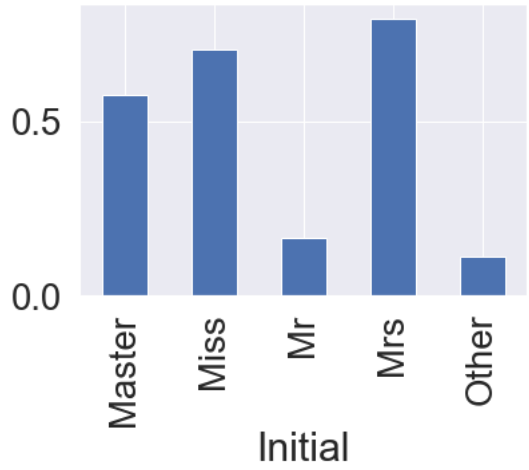
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
```

```
    text-align: right;
}
```

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare | FamilySize |
|---|---|---|---|---|---|---|---|---|
| **Initial** | | | | | | | | |
| **Master** | 414.975000 | 0.575000 | 2.625000 | 4.574167 | 2.300000 | 1.375000 | 3.340710 | 4.675000 |
| **Miss** | 411.741935 | 0.704301 | 2.284946 | 21.860000 | 0.698925 | 0.537634 | 3.123713 | 2.236559 |
| **Mr** | 455.880907 | 0.162571 | 2.381853 | 32.739609 | 0.293006 | 0.151229 | 2.651507 | 1.444234 |
| **Mrs** | 456.393701 | 0.795276 | 1.984252 | 35.981818 | 0.692913 | 0.818898 | 3.443751 | 2.511811 |
| **Other** | 564.444444 | 0.111111 | 1.666667 | 45.888889 | 0.111111 | 0.111111 | 2.641605 | 1.222222 |

```
df_train.groupby('Initial')['Survived'].mean().plot.bar()
```

```
<AxesSubplot:xlabel='Initial'>
```




```
df_train.groupby('Initial').mean()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare | FamilySize |
|---|---|---|---|---|---|---|---|---|
| **Initial** | | | | | | | | |
| **Master** | 414.975000 | 0.575000 | 2.625000 | 4.574167 | 2.300000 | 1.375000 | 3.340710 | 4.675000 |
| **Miss** | 411.741935 | 0.704301 | 2.284946 | 21.860000 | 0.698925 | 0.537634 | 3.123713 | 2.236559 |
| **Mr** | 455.880907 | 0.162571 | 2.381853 | 32.739609 | 0.293006 | 0.151229 | 2.651507 | 1.444234 |
| **Mrs** | 456.393701 | 0.795276 | 1.984252 | 35.981818 | 0.692913 | 0.818898 | 3.443751 | 2.511811 |
| **Other** | 564.444444 | 0.111111 | 1.666667 | 45.888889 | 0.111111 | 0.111111 | 2.641605 | 1.222222 |

```
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Mr'),'Age'] = 33
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Mrs'),'Age'] = 36
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Master'),'Age'] = 5
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Miss'),'Age'] = 22
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Other'),'Age'] = 46
```

```
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Mr'),'Age'] = 33
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Mrs'),'Age'] = 36
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Master'),'Age'] = 5
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Miss'),'Age'] = 22
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Other'),'Age'] = 46
```

```
print(f'Embarked has f{sum(df_train["Embarked"].isnull())} Null values')
```

```
Embarked has f2 Null values
```

```
df_train['Embarked'].fillna('S', inplace=True)
```

## 1.2 Change Age(Continuous to categorical)

```
def category_age(x):
    if x < 10:
        return 0
    elif x < 20:
        return 1
    elif x < 30:
        return 2
    elif x < 40:
        return 3
    elif x < 50:
        return 4
    elif x < 60:
        return 5
    elif x < 70:
        return 6
    else:
        return 7

df_train['Age_cat'] = df_train['Age'].apply(category_age)
df_test['Age_cat'] = df_test['Age'].apply(category_age)
```

```
df_train.drop(['Age'], axis=1, inplace=True)
df_test.drop(['Age'], axis=1, inplace=True)
```

## 1.3 Change Initial, Embarked and Sex(String to Numerical)

```
df_train['Initial'] = df_train['Initial'].map({'Master': 0, 'Miss': 1, 'Mr': 2, 'Mrs': 3, 'Other': 4})
df_test['Initial'] = df_test['Initial'].map({'Master': 0, 'Miss': 1, 'Mr': 2, 'Mrs': 3, 'Other': 4})
```

```
df_train['Embarked'].unique()
```

```
array(['S', 'C', 'Q'], dtype=object)
```

```
df_train['Embarked'].value_counts()
```

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

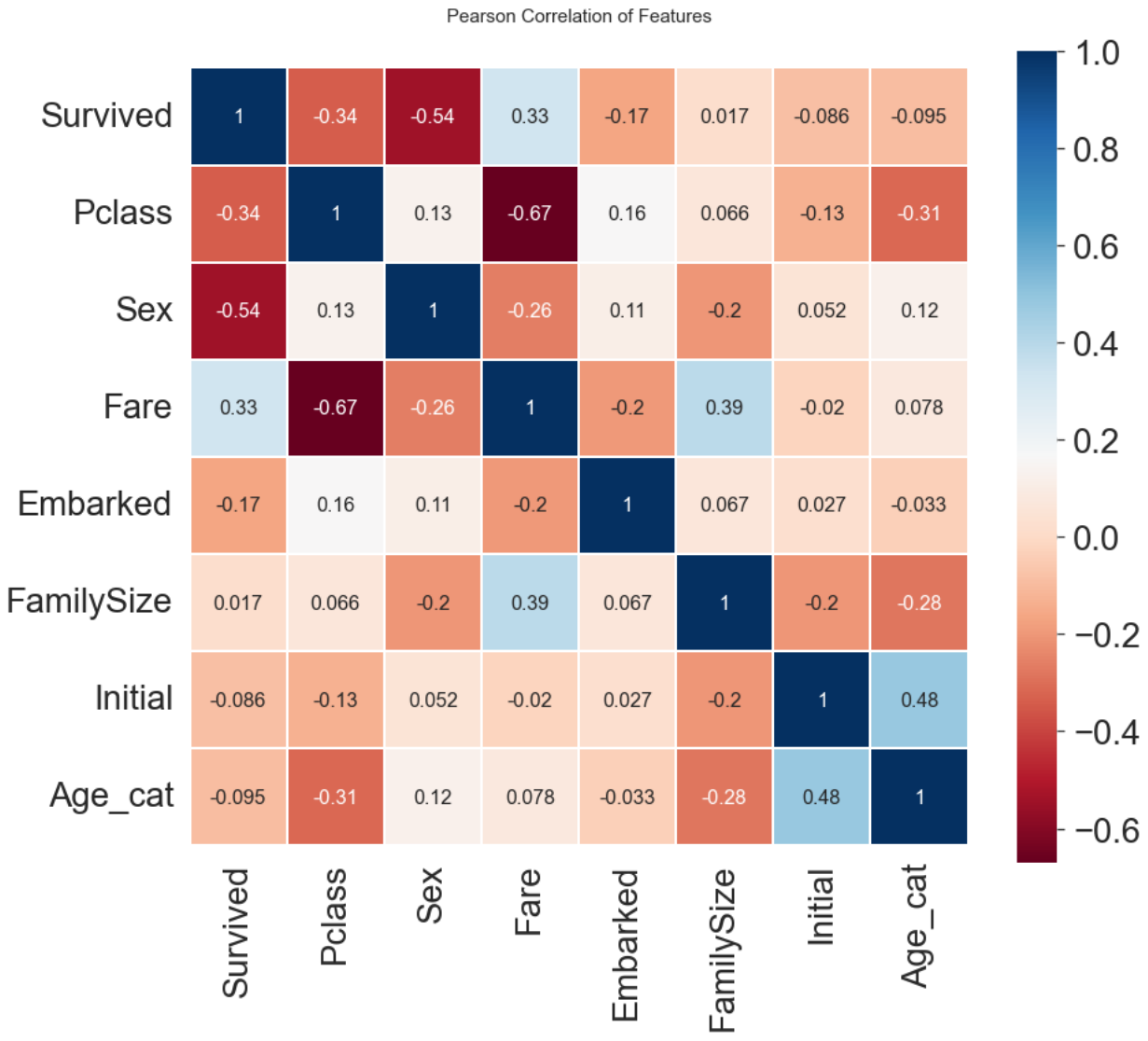
```
df_train['Embarked'] = df_train['Embarked'].map({'C': 0, 'Q': 1, 'S': 2})
df_test['Embarked'] = df_test['Embarked'].map({'C': 0, 'Q': 1, 'S': 2})
```

```
df_train['Embarked'].isnull().any()
```

```
False
```

```
df_train['Sex'] = df_train['Sex'].map({'female': 0, 'male': 1})
df_test['Sex'] = df_test['Sex'].map({'female': 0, 'male': 1})
```

```
heatmap_data = df_train[['Survived', 'Pclass', 'Sex', 'Fare', 'Embarked', 'FamilySize', 'Initial', 'Age_cat']]

colormap = plt.cm.RdBu
plt.figure(figsize=(14, 12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(heatmap_data.astype(float).corr(), linewidths=0.1, vmax=1.0,
            square=True, cmap=colormap, linecolor='white', annot=True, annot_kws={"size": 16})

del heatmap_data
```




## 1.4 One-hot encoding on Initial and Embarked

```
df_train = pd.get_dummies(df_train, columns=['Initial'], prefix='Initial')
df_test = pd.get_dummies(df_test, columns=['Initial'], prefix='Initial')
```

```
df_train.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| | PassengerId | Survived | Pclass | Name | Sex | SibSp | Parch | Ticket | Fare | Cabin | Embarked | FamilySize | Age_cat | Initial_0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | 1 | 1 | 0 | A/5 21171 | 1.981001 | NaN | 2 | 2 | 2 | 0 |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | 0 | 1 | 0 | PC 17599 | 4.266662 | C85 | 0 | 2 | 3 | 0 |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | 0 | 0 | 0 | STON/O2. 3101282 | 2.070022 | NaN | 2 | 1 | 2 | 0 |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | 0 | 1 | 0 | 113803 | 3.972177 | C123 | 2 | 2 | 3 | 0 |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | 1 | 0 | 0 | 373450 | 2.085672 | NaN | 2 | 1 | 3 | 0 |

```
df_train = pd.get_dummies(df_train, columns=['Embarked'], prefix='Embarked')
df_test = pd.get_dummies(df_test, columns=['Embarked'], prefix='Embarked')
```

## 1.5 Drop columns

```
df_train.drop(['Name', 'SibSp', 'Parch', 'Ticket', 'Cabin'], axis=1, inplace=True)
df_test.drop(['Name',  'SibSp', 'Parch', 'Ticket', 'Cabin'], axis=1, inplace=True)
```

```
df_train.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| | PassengerId | Survived | Pclass | Sex | Fare | FamilySize | Age_cat | Initial_0 | Initial_1 | Initial_2 | Initial_3 | Initial_4 | Embarked_0 | Emb |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | 1 | 1.981001 | 2 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| | PassengerId | Survived | Pclass | Sex | Fare | FamilySize | Age_cat | Initial_0 | Initial_1 | Initial_2 | Initial_3 | Initial_4 | Embarked_0 | Emb |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 2 | 1 | 1 | 0 | 4.266662 | 2 | 3 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| **2** | 3 | 1 | 3 | 0 | 2.070022 | 1 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **3** | 4 | 1 | 1 | 0 | 3.972177 | 2 | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **4** | 5 | 0 | 3 | 1 | 2.085672 | 1 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

```
df_test.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| | PassengerId | Pclass | Sex | Fare | FamilySize | Age_cat | Initial_0 | Initial_1 | Initial_2 | Initial_3 | Initial_4 | Embarked_0 | Embarked_1 | E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 892 | 3 | 1 | 2.057860 | 1 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **1** | 893 | 3 | 0 | 1.945910 | 2 | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **2** | 894 | 2 | 1 | 2.270836 | 1 | 6 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **3** | 895 | 3 | 1 | 2.159003 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| **4** | 896 | 3 | 0 | 2.508582 | 3 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

# 2. Machine Learning

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

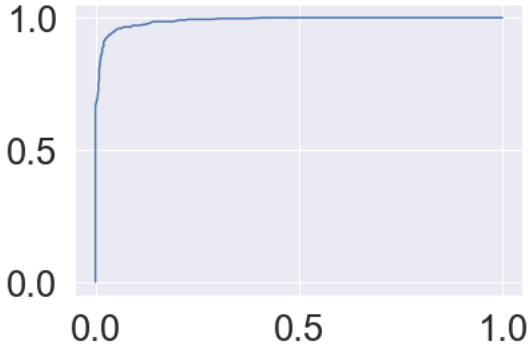
## 2.1 Create data for training

```
X_train = df_train.drop(['PassengerId','Survived'], axis=1)
Y_train = df_train['Survived']
X_test = df_test.drop(['PassengerId'], axis=1)
```

## 2.2 Classifier and accuracy, AUC, and ROC curve

```
classifier = RandomForestClassifier()
classifier.fit(X_train, Y_train)
accuracy = classifier.score(X_train, Y_train) * 100
Y_train_pred = classifier.predict_proba(X_train)[:, 1]

FPR, TPR, thresholds = roc_curve(Y_train, Y_train_pred)
AUC = roc_auc_score(Y_train, Y_train_pred)

plt.plot(FPR, TPR)
print("Accuracy: ", "{0:.2f}".format(accuracy))
print("Area Under the Curve: ", "{0:.2f}".format(AUC))
```

```
Accuracy:  95.40
Area Under the Curve:  0.99
```




## 2.3 Generate Kaggle submission

```
predict = classifier.predict(X_test)
predict = np.round(predict)

submission = pd.DataFrame({'PassengerId': df_test['PassengerId'], 'Survived': predict})
submission.to_csv('titanic/submission.csv', index=False)
```