```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from  sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
file_path='/content/drive/MyDrive/Titanic-Dataset.csv'
```

```
titanic_test=pd.read_csv(file_path)
```

```
print(titanic_test.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
None
```

```
titanic_test_list= list(titanic_test.columns)
print(titanic_test_list)
```

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

```
titanic_test.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC 17599	71.2

Next steps: Generate code with titanic_test View recommended plots

```
titanic_test.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fa
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.0000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.2042
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.6934
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.0000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.9104
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.4542
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.0000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.3292

```
print(titanic_test.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
## Now we will handle the missing values
titanic_test['Age'].fillna(titanic_test['Age'].median(), inplace=True)
```

```
titanic_test['Embarked'].fillna(titanic_test['Embarked'].mode()[0], inplace=True)
```

```
## Since Cabin no. is not useful we will drop it
titanic_test.drop(columns=['Cabin'], inplace=True)
```

```
print('Now dataset is cleaned\n', titanic_test.isnull().sum())
```

```
Now dataset is cleaned
 PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

```
# Sex and Embarked has categorical values so we will treat them by encoding with dummy variables
titanic_test= pd.get_dummies(titanic_test, columns=['Sex', 'Embarked'], drop_first=True)
```

```
# Since we dont need Name, Ticket and Passenger ID for prediction purpose we will drop them from our data set
titanic_test.drop(columns=['Name', 'Ticket', 'PassengerId'], inplace=True)
```

```
print("Our Final Dataset is-")
titanic_test.head()
```

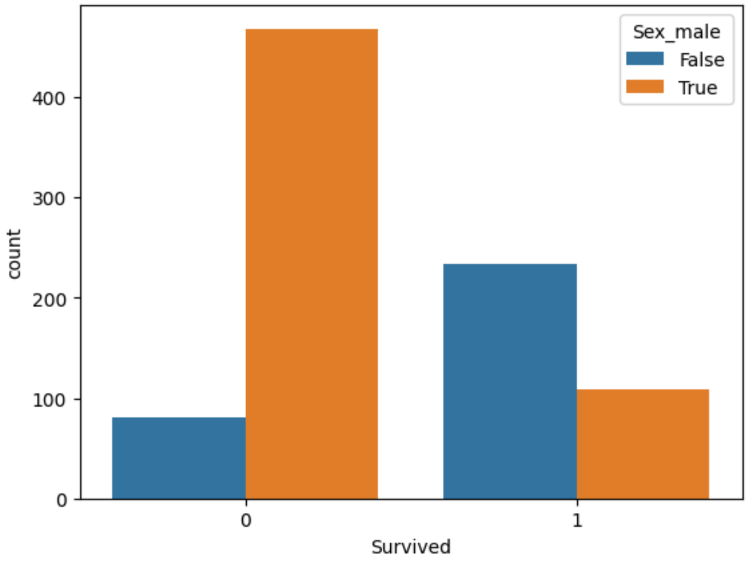
```
Our Final Dataset is-
```

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	0	3	22.0	1	0	7.2500	True	False	True
1	1	1	38.0	1	0	71.2833	False	False	False
2	1	3	26.0	0	0	7.9250	False	False	True
3	1	1	35.0	1	0	53.1000	False	False	True
4	0	3	35.0	0	0	8.0500	True	False	True

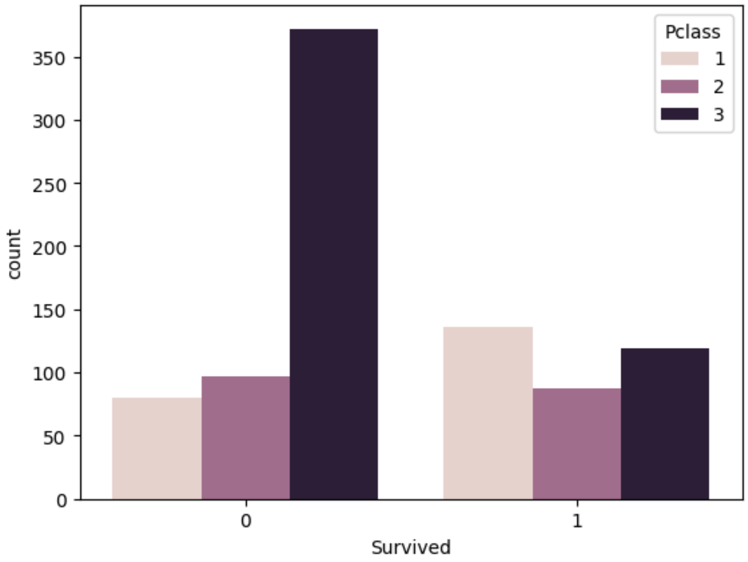
Next steps: Generate code with titanic_test View recommended plots

```
# Now we will visualise our survived and not survived person data set as-
print("No. of survived(1) and non-survived(0) people according to sex-")
sns.countplot(x='Survived', hue='Sex_male', data=titanic_test)
plt.show()
print("No. of survived(1) and non-survived(0) people according to class-")
sns.countplot(x='Survived', hue='Pclass', data=titanic_test)
plt.show()
print("No. of survived(1) and non-survived(0) people according to age-")
sns.kdeplot(titanic_test[titanic_test['Survived'] == 0]['Age'], label='Not Survived')
sns.kdeplot(titanic_test[titanic_test['Survived'] == 1]['Age'], label='Survived')
plt.legend()
plt.show()
```

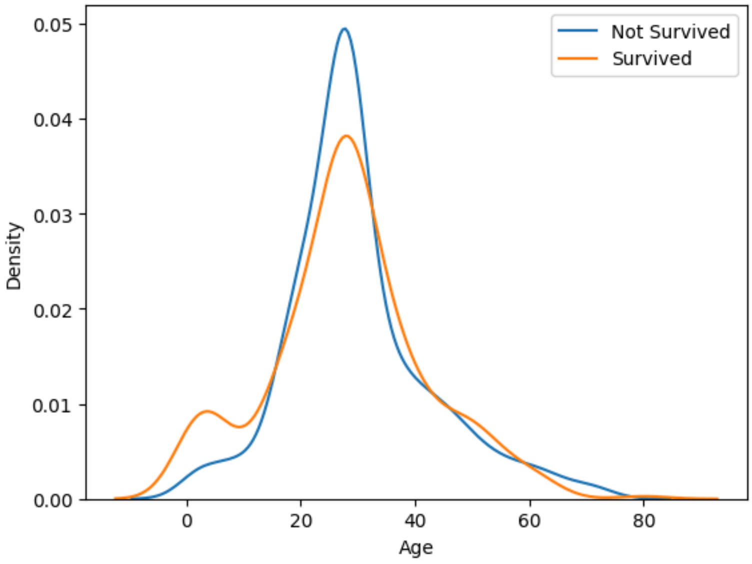
```
No. of survived(1) and non-survived(0) people according to sex-
```



```
No. of survived(1) and non-survived(0) people according to class-
```



```
No. of survived(1) and non-survived(0) people according to age-
```



```
features=list(set(titanic_test.columns)-set(['Survived']))
```

```
print(features)
```

```
['Embarked_Q', 'Pclass', 'Parch', 'SibSp', 'Age', 'Sex_male', 'Fare', 'Embarked_S']
```

```
target=(['Survived'])
```

```
print(target)
```

```
['Survived']
```

```
y=titanic_test[target].values
```

```
X=titanic_test[features].values
```

```
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.3,random_state=0)
```

```
scaler=StandardScaler()
```

```
scaler.fit(train_X)
```

```
▾ StandardScaler
StandardScaler()
```

```
train_X=scaler.transform(train_X)
```

```
test_X=scaler.transform(test_X)
```

LOGISTIC REGRESSION OF THE SAMPLE

```
Log_model=LogisticRegression()
```

```
Log_model.fit(train_X,train_y.ravel())
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_log_pred=Log_model.predict(test_X)
```

```
CM_log=confusion_matrix(y_log_pred,test_y)
```

```
print(CM_log)
```

```
[[141  28]
 [ 27  72]]
```

```
accuracy_log=accuracy_score(y_log_pred,test_y)
print("Accuracy score of the prediction by Logistic Regression Algorithm is", accuracy_log ,'.\n Logistic regression model of the sample
```

```
Accuracy score of the prediction by Logistic Regression Algorithm is 0.7947761194029851 .
 Logistic regression model of the sample is 79.47761194029852 % accurate.
```

K-NEAREST NEIGHBORS CLASSIFIER

```
KNN = KNeighborsClassifier(n_neighbors=2)
```

```
KNN.fit(train_X,train_y.ravel())
```

```
▾       KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

```
KNN_pred=KNN.predict(test_X)
```

```
confusion_matrix=confusion_matrix(test_y,KNN_pred)
```

```
print(confusion_matrix)
```

```
[[154  14]
 [ 44  56]]
```

```
accuracy_score_KNN=accuracy_score(KNN_pred,test_y)
print("Accuracy score of the prediction by K-Nearest Neighbors Classifier Algorithm is", accuracy_score_KNN ,'.\nK-Nearest Neighbors Cla
```

```
Accuracy score of the prediction by K-Nearest Neighbors Classifier Algorithm is 0.7835820895522388 .
K-Nearest Neighbors Classifier model of the sample is 78.35820895522389 % accurate.
```

SUPPORT VECTOR MACHINE

```
clf=SVC(kernel='linear')
```

```
clf.fit(train_X,train_y.ravel())
```

```
▾        SVC
SVC(kernel='linear')
```

```
SVM_pred=clf.predict(test_X)
```

```
accuracy_SVM=accuracy_score(test_y, SVM_pred)
```

```
print("Accuracy score of the prediction by Support Vector Machine Algorithm is", accuracy_SVM ,'.\n Support Vector Machine model of the
```

```
Accuracy score of the prediction by Support Vector Machine Algorithm is 0.7873134328358209 .
 Support Vector Machine model of the sample is 78.73134328358209 % accurate.
```

DECISION TREE CLASSIFIER

```
dt_classifier = DecisionTreeClassifier(criterion='gini')
```

```
dt_classifier.fit(train_X, train_y)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred_dt = dt_classifier.predict(test_X)
```

```
accuracy_dt = accuracy_score(test_y, pred_dt)
print("Accuracy score of the prediction by Decision Tree Classifier Algorithm is", accuracy_dt ,'.\n Decision Tree Classifier model of t
```

```
Accuracy score of the prediction by Decision Tree Classifier Algorithm is 0.7835820895522388 .
 Decision Tree Classifier model of the sample is 78.35820895522389 % accurate.
```

RANDOM FOREST CLASSIFIER

```
Classifier= RandomForestClassifier(random_state=90)
```

```
Classifier.fit(train_X,train_y.ravel())
```

```
▾        RandomForestClassifier
RandomForestClassifier(random_state=90)
```

```
y_pred = Classifier.predict(test_X)
```

```
params= {'max_depth':[15,20,25],
       'max_features':['auto','sqrt'],
       'min_samples_split':[15,20,25],
       'min_samples_leaf':[5,10],
       'n_estimators':[10,25,30]}
```