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


titanic=pd.read_csv("/kaggle/input/titanic-survival/Titanic-Dataset.csv")


#Default theme
sns.set_theme(context='notebook',
              style='whitegrid',
              palette='rainbow',
              font='Lucida Calligraphy',
              font_scale=1.5,
              rc=None)
import matplotlib
matplotlib.rcParams['figure.figsize'] = [4, 4]
matplotlib.rcParams.update({'font.size': 15})
matplotlib.rcParams['font.family'] = 'sans-serif'


titanic.head()


titanic.tail()


titanic.info()


titanic.describe()


matplotlib.rcParams.update({'font.size': 10})

titanic.dtypes.value_counts().plot.pie(explode=[0.1, 0.1, 0.1],
                                       autopct='%1.2f%%',
                                       shadow=True)
plt.title('Data Type',
          color='Green',
          loc='center',
          font='Lucida Calligraphy');


ax = sns.set(style="whitegrid")
ax = sns.countplot(data=titanic,x='Embarked');
ax.bar_label(ax.containers[0])

plt.title('Embarked Distribution',color='Red',loc='center',font='Lucida Calligraphy');
plt.xlabel('Embarked',color='Green',loc='center',font='Lucida Calligraphy')
plt.ylabel('Count',color='Red',loc='center',font='Lucida Calligraphy');


matplotlib.rcParams.update({'font.size': 10})
titanic['Embarked'].value_counts().plot.pie(explode=[0.1, 0.1, 0.1],
                                       autopct='%1.2f%%',
                                       shadow=True)
plt.title('Embarked Distribution',color='Red',loc='center');


matplotlib.rcParams.update({'font.size': 10})
titanic['Sex'].value_counts().plot.pie(explode=[0.1, 0.1],
                                       autopct='%1.2f%%',
                                       shadow=True)
plt.title('Gender Distribution',color='Red',loc='center');


matplotlib.rcParams.update({'font.size': 10})
titanic['Survived'].value_counts().plot.pie(explode=[0.1, 0.1],
                                       autopct='%1.2f%%',
                                       shadow=True)
plt.title('Survived',color='Red',loc='center');


## Combining Data
titanic.agg(
    {
        "Fare": ["min", "max", "median", "mean","skew",'std'],
        "Age": ["min", "max", "median", "mean","skew",'std'],
    }
)
```

```
titanic_fig = sns.FacetGrid(titanic, col='Sex', hue='Survived',height=6,aspect=1.6)
titanic_fig.map(plt.scatter,'Fare','Age' )
plt.show()


plt.figure(figsize=(8,3))

titanic.nunique().plot(kind='bar')
plt.title('No of unique values in the dataset')
plt.show()


sns.pairplot(titanic,vars=['Age','Fare','Survived'],hue='Sex',palette='plasma',aspect=1.9);


numerical = titanic.select_dtypes(include=['number']).columns
categorical = titanic.select_dtypes(include=['object']).columns

print('Numerical  :',numerical)
print("**********"*10)
print("Categorical:",categorical)


for col in titanic[['Sex', 'Embarked','Survived','Pclass','SibSp','Parch']]:
    print(titanic[col].value_counts())
    print("****"*7)


titanic.isnull().mean().sort_values(ascending=False)*100


def missing_value (df):
    missing_Number = df.isnull().sum().sort_values(ascending=False)[df.isnull().sum().sort_values(ascending=False) !=0]
    missing_percent=round((df.isnull().sum()/df.isnull().count())*100,2)[round((df.isnull().sum()/df.isnull().count())*100,2) !=0]
    missing = pd.concat([missing_Number,missing_percent],axis=1,keys=['Missing Number','Missing Percentage'])
    return missing


missing_values = titanic.isnull().sum()
missing_values = missing_values[missing_values > 0]
missing_values.sort_values(inplace=True)
missing_values.plot.pie(explode=[0.1, 0.1, 0.1],
                        autopct='%1.2f%%',
                        shadow=True)

plt.title('Missing Values',
          color='Green',
          loc='center',
          font='Lucida Calligraphy');


sns.heatmap(titanic.isnull(),cmap='cool');


import missingno as msno
msno.matrix(titanic)
plt.show()


import missingno
missingno.bar(titanic, color="dodgerblue", sort="ascending", figsize=(10,5), fontsize=12);


missingno.matrix(titanic, figsize=(10,5), fontsize=12, color=(1, 0.38, 0.27));
```

Filling/Removing Missing Values

```
titanic['Age'] = titanic['Age'].fillna(titanic['Age'].mean())


titanic[titanic['Embarked_Q'].isnull()]
titanic[titanic['Embarked_S'].isnull()]


titanic['Embarked_Q'] = titanic['Embarked_Q'].fillna(method='bfill')
titanic['Embarked_S'] = titanic['Embarked_S'].fillna(method='bfill')


titanic = titanic.drop(['Cabin'],axis=1)


import missingno as msno
msno.matrix(titanic)
plt.show()
```

```
titanic.isnull().sum()
```

Filling/Removing Missing Values

```
titanic = titanic.drop(['Name','Ticket'],axis=1)
```

```
titanic.head()
```

Categorical feature (Sex & Embarked) is converted into numerical feature by using pandas dummy method

```
titanic = pd.get_dummies(titanic,columns=['Sex','Embarked'],drop_first=True)
titanic.head()
```

Train Test Split

```
X = titanic.drop(['Survived'],axis=1)
y = titanic['Survived']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=21)
```

#Standardizing the data

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)
```

```
display(X_train.head())
display(X_test.head())
```

Model Implementation

```
#Logistic Regression
```

```
from sklearn.metrics import accuracy_score
# Logistic Regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)

log_train = round(logreg.score(X_train, y_train) * 100, 2)
log_accuracy = round(accuracy_score(Y_pred, y_test) * 100, 2)

print("Training Accuracy    :",log_train)
print("Model Accuracy Score :",log_accuracy)
```

```
# Support Vector Machines
```

```
# Support Vector Machines
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
Y_pred = svc.predict(X_test)

svc_train = round(svc.score(X_train, y_train) * 100, 2)
svc_accuracy = round(accuracy_score(Y_pred, y_test) * 100, 2)

print("Training Accuracy     :",svc_train)
print("Model Accuracy Score  :",svc_accuracy)
```

KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
Y_pred = knn.predict(X_test)

knn_train = round(knn.score(X_train, y_train) * 100, 2)
knn_accuracy = round(accuracy_score(Y_pred, y_test) * 100, 2)

print("Training Accuracy     :",knn_train)
print("Model Accuracy Score  :",knn_accuracy)
```

Random forest classifier

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, y_train)

random_forest_train = round(random_forest.score(X_train, y_train) * 100, 2)
random_forest_accuracy = round(accuracy_score(Y_pred, y_test) * 100, 2)

print("Training Accuracy     :",random_forest_train)
print("Model Accuracy Score  :",random_forest_accuracy)
```