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

In [ ]:
```
train = pd.read_csv('C:/Users/banke/Desktop/TT/titanic_train.csv')
```

In [ ]:
```
train.head()
```

In [ ]:
```
train.info()
```

In [ ]:
```
sns.heatmap(train.isnull(), yticklabels=False, cmap ='viridis')
```

In [ ]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived', hue ='Sex', data=train, palette='RdBu_r')
```

In [ ]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Pclass', data=train,palette="rainbow")
```

In [ ]:
```
plt.figure(figsize=(12,7))
sns.boxplot(x='Pclass', y='Age',data=train, palette = 'winter')
```

In [ ]:
```
def impute_age(cols):
    Age = cols[0]
    Pclass=cols[1]

    if pd.isnull (Age):
        if Pclass ==1:
            return 37
        elif Pclass == 2:
            return 29
        else :
            return 24
    else:
        return Age
```

In [ ]:
```
train['Age'] = train[['Age', 'Pclass']].apply(impute_age, axis=1)
```

In [ ]:
```
sns.heatmap(train.isnull(), yticklabels=False, cmap='viridis')
```

In [ ]:
```
train.drop('Cabin',axis = 1, inplace =True)
train.dropna(inplace=True)
train.head()
```

In [ ]:
```
train.info()
```

In [ ]:
```
sex= pd.get_dummies(train['Sex'], drop_first=True)
embark = pd.get_dummies(train['Embarked'], drop_first=True)
```

In [ ]:
```
train.drop(['Sex', 'Embarked', 'Name', 'Ticket'], axis = 1, inplace=True)
```

In [ ]:
```
train = pd.concat([train, sex, embark], axis = 1)
```

In [ ]:
```
train.head(10)
```

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(train.drop('Survived',axis=1),
                                                    train['Survived'],
                                                    test_size = 0.3,
                                                    random_state=101)
```

In [ ]:
```
#Training and Predicting

from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression()

logmodel.fit (X_train, y_train)
```

In [ ]:
```
predictions = logmodel.predict(X_test)
```

In [ ]:
```
#Evaluation

from sklearn import metrics

print ("Accuracy: ", metrics.accuracy_score(y_test,predictions))
print ("Precision: ", metrics.precision_score(y_test,predictions))
print ("Recall: ", metrics.recall_score(y_test,predictions))
```

In [ ]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, predictions)
print (cm)
```

In [ ]:
```
class_names = [0,1]

fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)

sns.heatmap(pd.DataFrame(cm), annot=True, cmap ='YlGnBu', fmt = 'g')
ax.xaxis.set_label_position("top")

plt.tight_layout()
plt.title("Confusion Matrix")
plt.ylabel("Actual label")
plt.xlabel("Predicted label")
```

In [ ]:
```
from sklearn.datasets import fetch_openml
X,y = fetch_openml('mnist_784', version = 1, return_X_y=True)
```

In [ ]:
```
X.shape
```

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test= train_test_split(X,
                                                   y,
                                                   test_size = 1/7,
                                                   random_state=0)
```

In [ ]:
```
X_train.shape
```

In [ ]:
```
y_test.shape
```

In [ ]:
```
X_test.shape
```

In [ ]:
```
plt.figure(figsize=(20,4))
for index in range (5):
    plt.subplot(1,5, index+1)
    plt.imshow(X_train.to_numpy()[index].reshape((28,28)), cmap=plt.cm.gray)
    plt.title ('Training : %i\n' % int(y_train.to_numpy()[index]), fontsize=20)
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression()

logmodel.fit (X_train, y_train)
```

In [ ]:
```
predictions = logmodel.predict(X_test)
```

In [ ]:
```
score = logmodel.score(X_test, y_test)
print (score)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, predictions)
```

In [ ]:
```
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt = ".2f", linewidth=0.5, square=True ,cmap ="Blues_r")
ax.xaxis.set_label_position("top")

plt.ylabel("Actual label")
plt.xlabel("Predicted label")
plt.title('Accuracy Score : {0}' .format(score), size=15)
plt.show()
```

In [ ]:
```

```