Precision and recall:

```
import numpy
from sklearn import metrics
import matplotlib.pyplot as plt
actual=numpy.random.binomial(1,0.9,size=1000)
predicted=numpy.random.binomial(1,0.9,size=1000)
Precision=metrics.precision_score(actual,predicted)
print(Precision)
Sensitivity_recall=metrics.recall_score(actual,predicted)
print(Sensitivity_recall)
```

Confusion matrix:

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics
actual=numpy.random.binomial(1,.9,size=1000)
predicted =numpy.random.binomial(1,.9,size =1000)
confusion_matrix =metrics.confusion_matrix(actual,predicted)
# Correct the typo in the keyword argument name
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix,display_labels = [False, True])
cm_display.plot()
plt.show()
```

Overfitting and underfitting:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
np.random.seed(42)
X = np.random.rand(100, 1)
y = 2 + 3 * X + np.random.randn(100, 1)
X_train, X_test, y_train, y_test =train_test_split(X, y,
test_size=0.2,random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_train_pred = model.predict(X_train)
mse_train = mean_squared_error(y_train,y_train_pred)
y_test_pred = model.predict(X_test)
mse_test = mean_squared_error(y_test, y_test_pred)

print(f"Training set MSE: {mse_train:.2f}")
print(f"Test set MSE: {mse_test:.2f}")
```

KNN:

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

dataset = pd.read_csv("Social_Network_Ads.csv")
x=dataset.iloc[:,[2,3]].values
y=dataset.iloc[:,4].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors =5 , metric ='minkowski')
classifier.fit(x_train,y_train)
print(classifier.score(x_test,y_test))
y_pred = classifier.predict(x_test)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

LDA:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
iris = load_iris()
dataset = pd.DataFrame(columns=iris.feature_names,data=iris.data)
dataset['target'] = iris.target
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, 4].values
sc = StandardScaler()
X = sc.fit_transform(X)
le = LabelEncoder()
y = le.fit_transform(y)
```

```
X_train, X_test,y_train, y_test = train_test_split(X, y,test_size=0.2)
lda = LinearDiscriminantAnalysis(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
plt.scatter(
    X_train[:, 0], X_train[:, 1],
    c=y_train,
    cmap='rainbow',
    alpha=0.7, edgecolors='b'
)
classifier = RandomForestClassifier(max_depth=2,random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print('Accuracy : ' + str(accuracy_score(y_test, y_pred)))
conf_m = confusion_matrix(y_test, y_pred)
print(conf_m)
```

K Means:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Mall_Customers.csv')
x = dataset.iloc[:, [3, 4]].values
from sklearn.cluster import KMeans
wcss_list = []
for i in range(1, 11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(x)
wcss_list.append(kmeans.inertia_)
plt.plot(range(1, len(wcss_list) + 1), wcss_list)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('WCSS_list')
plt.show()
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)
y_predict = kmeans.fit_predict(x)
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s=100, c='red')
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s=100,
c='blue')
plt.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s=100,
c='green', label='Cluster 2')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s=300, c='yellow', label='Centroid')
plt.title
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
```

```
mtp.show()
```

Navies_Bayes:

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset = pd.read_csv('user_data.csv')
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =train_test_split(x, y, test_size =
0.25,random_state=0)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print(y_pred)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

SVM:

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
dataset = pd.read_csv("Social_Network_Ads.csv")
X = dataset.iloc[:, [2, 3]].values
Y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.25, random_state=0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
```

Logistic Regression:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import io
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn import preprocessing
plt.rc("font", size = 14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
sns.set(style="white")
sns.set(style="whitegrid", color_codes = True)
df = pd.read_csv('candy-data.csv')
df =
df[['fruity','caramel','peanutyalmondy','nougat','crispedricewafer','hard',
'bar','pluribus','sugarpercent','pricepercent','winpercent','chocolate']]
df.head()
from sklearn.model_selection import train_test_split
trainingSet, testSet = train_test_split(df, test_size=0.2)
train_df = trainingSet
test_df = testSet
X_train =
train_df[['fruity','caramel','peanutyalmondy','nougat','crispedricewafer','har
d', 'bar','pluribus','sugarpercent','pricepercent','winpercent']]
y_train = train_df["chocolate"]
X_test =
test_df[['fruity','caramel','peanutyalmondy','nougat','crispedricewafer','hard
', 'bar','pluribus','sugarpercent','pricepercent','winpercent']]
y_test = test_df["chocolate"]
y_test.head()
y_train.value_counts()
plt.show()
plt.savefig('count_plot')
count_no_choc = len(train_df[train_df['chocolate']==0])
count_choc = len(train_df[train_df['chocolate']==1])
pct_of_no_choc = count_no_choc/(count_no_choc+count_choc)
print("percentage of no chocolate is", pct_of_no_choc*100)
pct_of_choc = count_choc/(count_no_choc+count_choc)
print("percentage of chocolate", pct_of_choc*100)
train_df.groupby('chocolate').mean()
train_df.groupby('caramel').mean()
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train,y_train)
y_pred = logreg.predict(X_test)
```

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

Linear Regression:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
dataset = pd.read_csv("Salary_Data.csv")
x = np.array(np.array(dataset.iloc[:,-1].values, ndmin=2))
y = np.array(dataset.iloc[:,1].values)
x.shape, y.shape
x = x.reshape(-1,1)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
y_pred = regressor.predict(x_test)
def mse(actual, predicted):
    return np.mean((np.square(actual - predicted)))
mean_squared_error = mse(y_test, y_pred)
print("Mean Sqaured Error = {}".format(mean_squared_error))
print("Root Mean Sqaured Error(RMSE) ={}".format(mean_squared_error**0.5))
```

```
plt.scatter(x_train,y_train,color='red')
plt.plot(x_train,regressor.predict(x_train),color='blue')
plt.title('Salary vs Experience(Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
plt.scatter(x_test,y_test,color='red')
plt.plot(x_test,regressor.predict(x_test),color='blue')
plt.title('Salary vs Experience(Testing set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```