Linear Regression

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

Step 1: Split the data into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
scaler = MinMaxScaler() # Or we can use StandardScaler
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 2: Create a Linear Regression model:

```
lr = LinearRegression()
```

Step 3: Train the model on the training data:

```
lr.fit(X_train_scaled, y_train)
```

Step 4: Predict using the model:

```
y_pred = lr.predict(X_test_scaled)
```

Step 5: Evaluate the model:

```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

Logistic Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
```

Step 0: Load the data and create the variables:

```
X = predictors
y = target
y_num = LabelEncoder.fit_transform(y)
```

Step 1: Split the data into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y_num , test_size=0.2)

scaler = MinMaxScaler() # Or we can use StandardScaler
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 2: Create a Logistic Regression model:

```
log_reg = LogisticRegression()
```

Step 3: Train the model on the training data:

```
log_reg.fit(X_train_scaled, y_train)
```

Step 4: Predict using the model:

```
y_pred = log_reg.predict(X_test_scaled)
```

Step 5: Evaluate the model:

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Step 6: Compute and visualize the confusion matrix

```
cm = confusion_matrix(y_test, y_pred)
```

KNN

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

Step 0: Load the data and create the variables:

```
X = predictors
y = target
y_num = LabelEncoder.fit_transform(y)
```

Step 1: Split the data into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y_num , test_size=0.2)

scaler = MinMaxScaler() # Or we can use StandardScaler
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 2: Create a KNN classifier model

```
knn = KNeighborsClassifier(n_neighbors=5)  # Set k (number of neighbors)
```

Step 3: Train the model on the training data

```
knn.fit(X_train_scaled, y_train)
```

Step 4: Predict using the model

```
y_pred = knn.predict(X_test_scaled)
```

Step 5: Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Step 6: Compute and visualize the confusion matrix

```
cm = confusion_matrix(y_test, y_pred)
```

Note: you can use the documentation to find any method easily without memorizing anything!