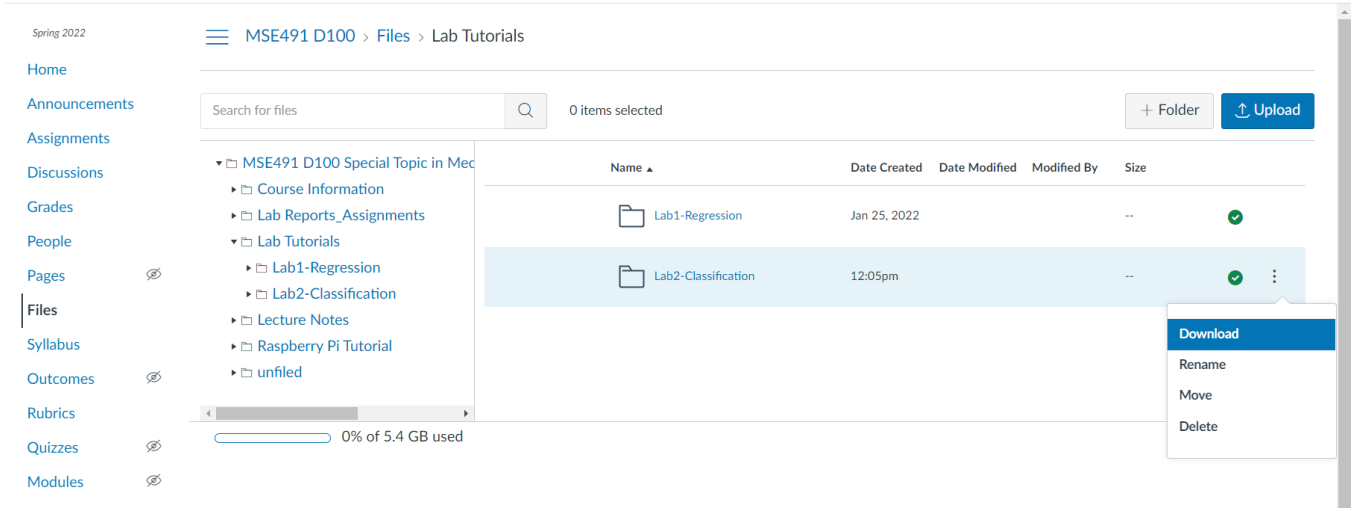
Simon Fraser University
School of Mechatronic Systems Engineering

MSE491 - Application of Machine Learning in Mechatronic Systems
Spring 2022

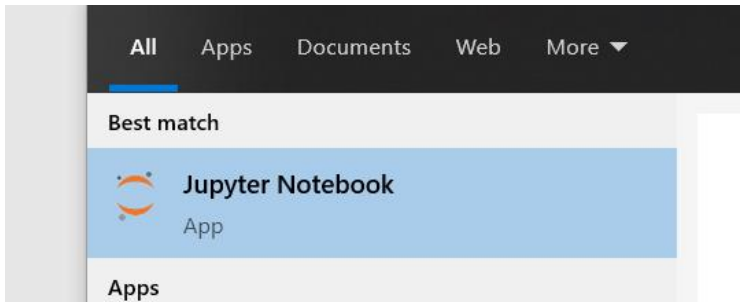
Lab 2 – Classification

Presenter: Samira Asadi
saa143@sfu.ca

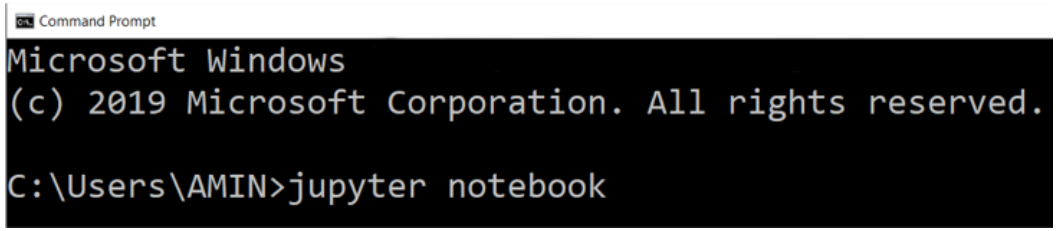
1. Download the lab tutorial.



2. Start the Jupyter notebook server.



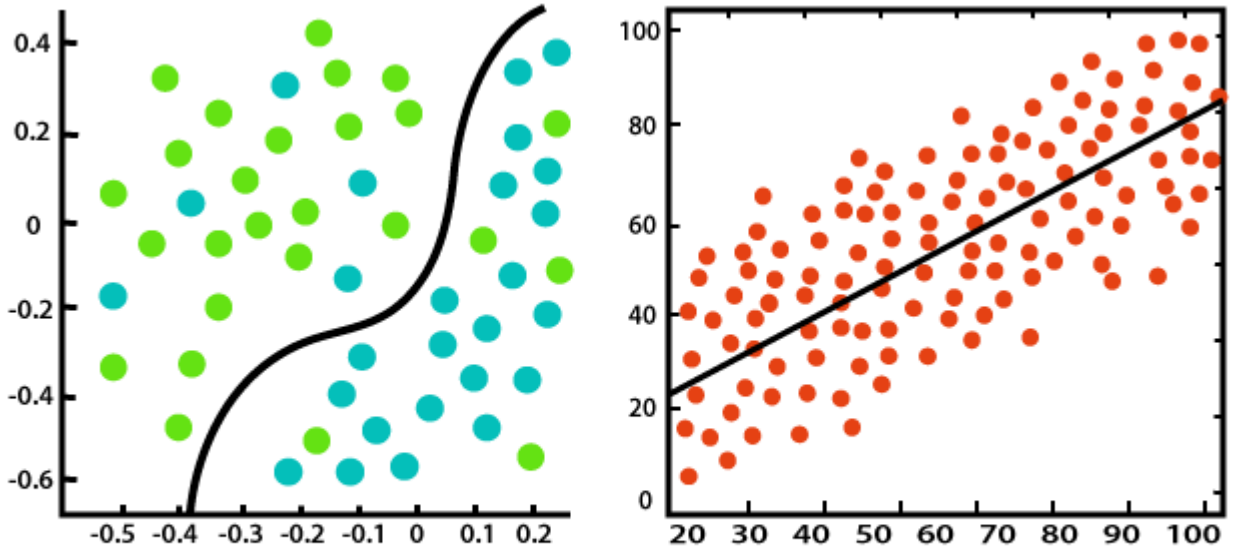
or



CLASSIFICATION vs. REGRESSION

Regression and classification are categorized under the same umbrella of supervised machine learning.

The main difference between these techniques is that the regression task's output variable is numerical (or continuous) while it is categorical (or discrete) for classification



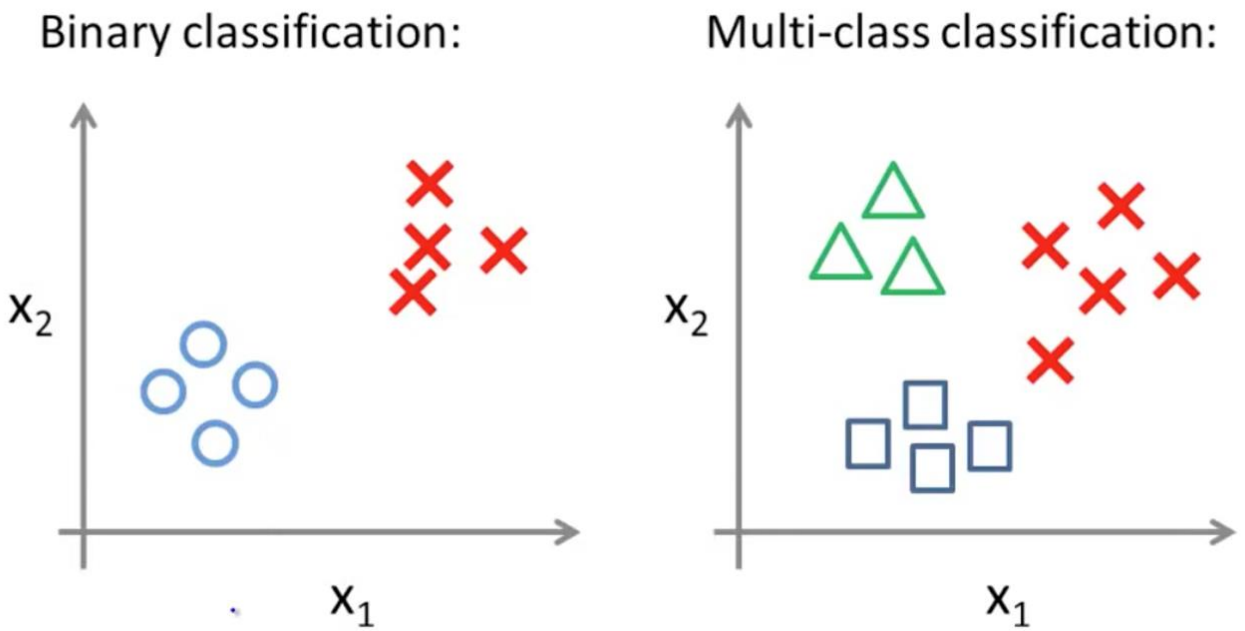
© https://www.javatpoint.com/regression-vs-classification-in-machine-learning

BINARY-CLASS CLASSIFICATION

In binary-class classifications, the given dataset is categorized into two categories.

MULTI-CLASS CLASSIFICATION

In multi-class classification, the given dataset is categorized into several (more than two) classes.

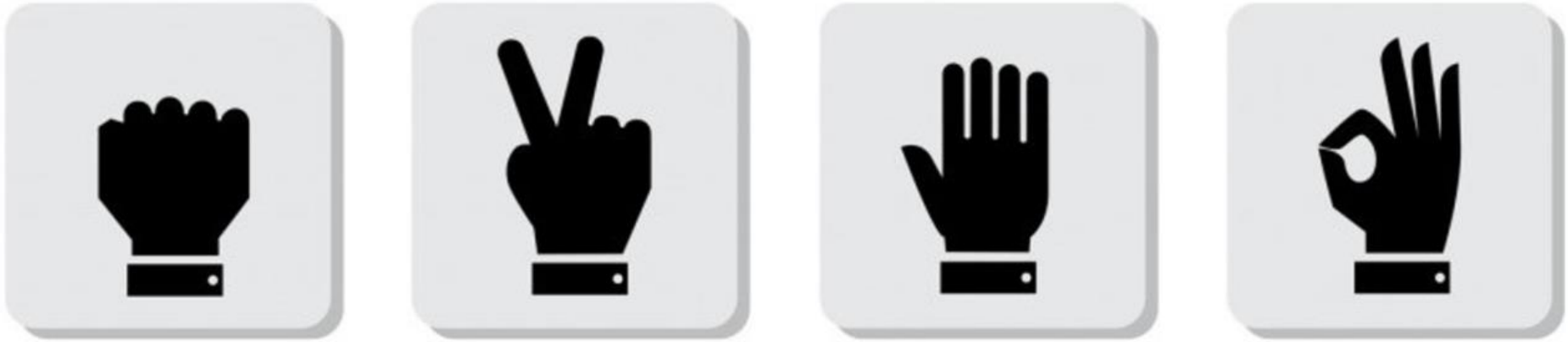


© https://medium.com/analytics-vidhya/logistic-regression-from-scratch-multi-classification-with-onevsall-d5c2acf0c37c

Classify hand gestures by reading muscle activity

https://www.kaggle.com/kyr7plus/emg-4

- Using Electromyography (EMG) to record the muscle movement activities when doing 4 different tasks.
- An armband with 8 sensors was placed on skin surface to measure the electrical activity produced by muscles beneath.
- Each dataset line has 8 consecutive readings of all 8 sensors.

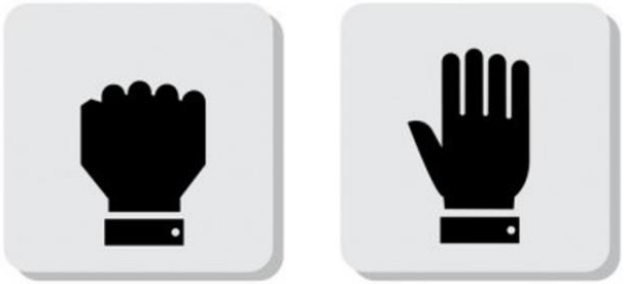


Recommended Video about Bionics:
https://www.youtube.com/watch?v=GgTwa3CPrIE

LOADING THE DATASET

```python
# Load General Libraries
import numpy as np
import pandas as pd
import pickle
import matplotlib.pyplot as plt

# Importing the dataset (no header)
df0 = pd.read_csv("EMG\EMG_0.csv", header=None)
df1 = pd.read_csv("EMG\EMG_1.csv", header=None)
df2 = pd.read_csv("EMG\EMG_2.csv", header=None)
df3 = pd.read_csv("EMG\EMG_3.csv", header=None)
dataset = pd.concat([df0,df1,df2,df3], axis = 0)

# Importing hand gestures images
import matplotlib.image as mpimg
img0 = mpimg.imread('EMG/0.jpg')
img1 = mpimg.imread('EMG/1.jpg')
img2 = mpimg.imread('EMG/2.jpg')
img3 = mpimg.imread('EMG/3.jpg')
```

BINARY CLASSIFICATION




```
# Creating a Binary Dataset (by only considering 2 labels)
# Class 0: Closed Fist
# Class 1: Open Hand

dataset_binary = dataset[dataset.iloc[:, -1].isin([0,2])]
dataset_binary.iloc[:, -1].replace(to_replace=2, value=1, inplace=True)

# Split features and targets - X: Features, y: Targets
X_binary = dataset_binary.iloc[:, :-1]
y_binary = dataset_binary.iloc[:, -1].values
```

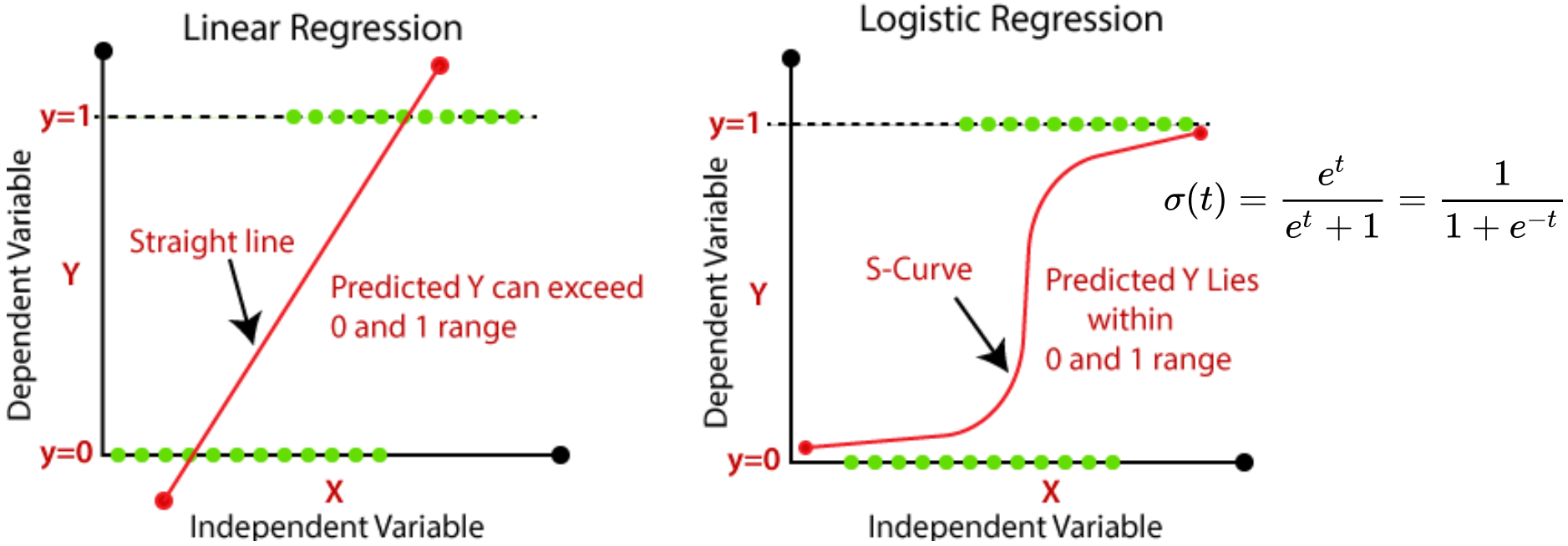
Test the difference between the mean values of two features from different classes

In a binary classification problem, where each sample can be classified either into class C1 or class C2, t-test helps us to evaluate whether the values of a particular feature for class C1 is significantly different from values of same feature for class C2. If this holds, then the feature can help us to better differentiate our data.

```
from scipy.stats import ttest_ind

FEATURES_C1 = dataset_binary[dataset_binary.iloc[:, -1]==0] # features for closed fist
FEATURES_C2 = dataset_binary[dataset_binary.iloc[:, -1]==1] # features for open hand

# Test the significance
T_STATISTIC, P_VALUE = ttest_ind(FEATURES_C1.iloc[:, 0],FEATURES_C2.iloc[:, 0])

print("p-value =", P_VALUE)
# if the p-value is less than 0.05 we have significant different between feature values
if P_VALUE < 0.05:
    print("significant")
else:
    print("not significant")
```

LOGISTIC REGRESSION

Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems.



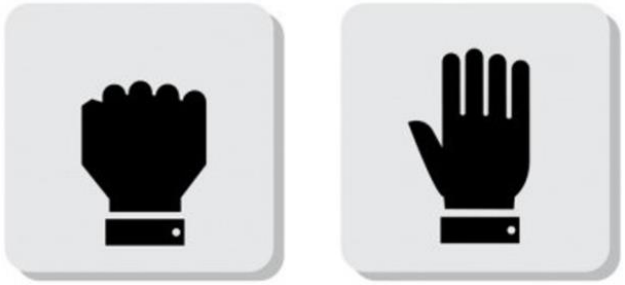
© https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning

https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

LOGISTIC REGRESSION

`sklearn.linear_model.LogisticRegression`

BINARY CLASSIFICATION



0 - Closed Fist

1 - Open Hand

```python
# Import the class
from sklearn.linear_model import LogisticRegression

# Instantiating the model (using the default parameters)
logreg = LogisticRegression()

# Fitting the model with data
logreg.fit(X_train_binary,y_train_binary)

# Predicting the labels on test set
y_pred_binary = logreg.predict(X_test_binary)
```

EVALUATING CLASSIFICATION PERFORMANCE CONFUSION MATRIX

A confusion matrix is a summary of prediction results on a classification problem.

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix with 4 values:

- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

EVALUATING CLASSIFICATION PERFORMANCE

CONFUSION MATRIX

`sklearn.metrics`.confusion_matrix

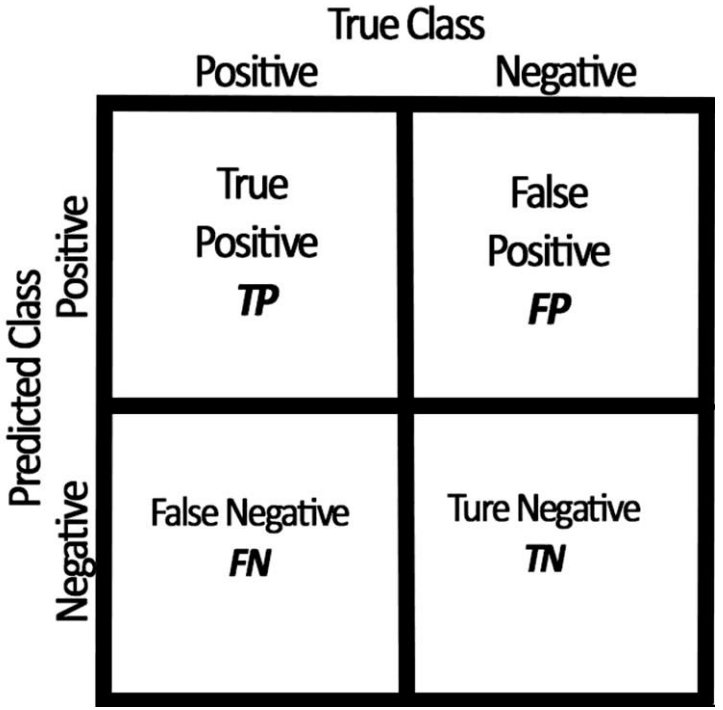
`sklearn.metrics`.plot_confusion_matrix

True Positive (TP)
The actual value was positive, and the model predicted a positive value

True Negative (TN)
The actual value was negative, and the model predicted a negative value

False Positive (FP) – Type 1 error
The actual value was negative, but the model predicted a positive value

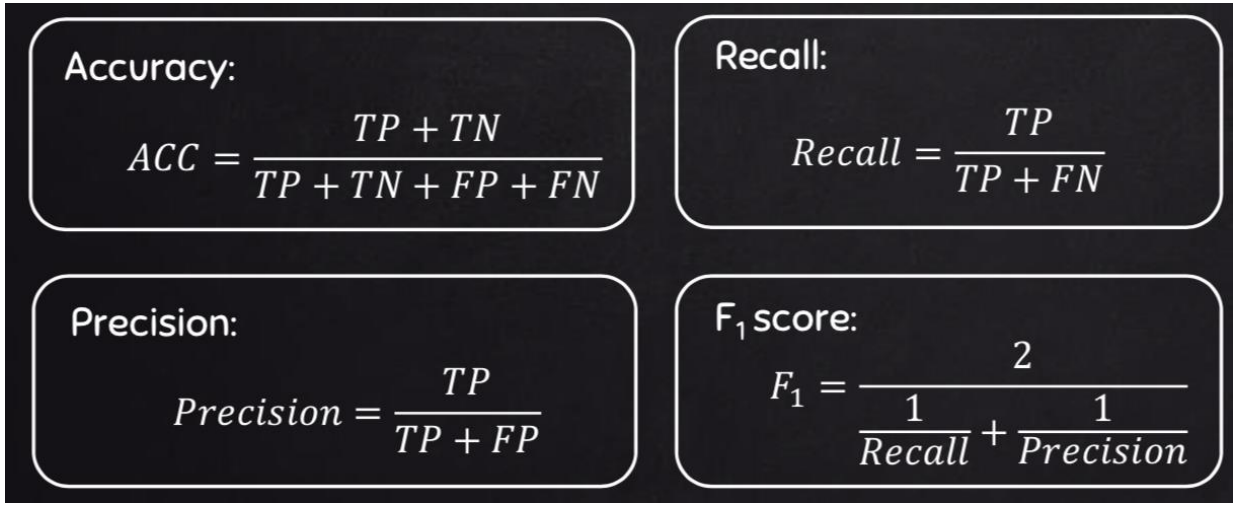
False Negative (FN) – Type 2 error
The actual value was positive, but the model predicted a negative value



Evaluation of the performance of a binary classification model using the confusion matrix

EVALUATING CLASSIFICATION PERFORMANCE

- **Precision** is the ratio of True Positives to all the positives predicted by the model.
- **Recall** (Sensitivity) is the ratio of True Positives to all the positives in your Dataset.
- **F-Measure** provides a single score that balances both the concerns of precision and recall in one number.



`sklearn.metrics`.accuracy_score

`sklearn.metrics`.precision_score

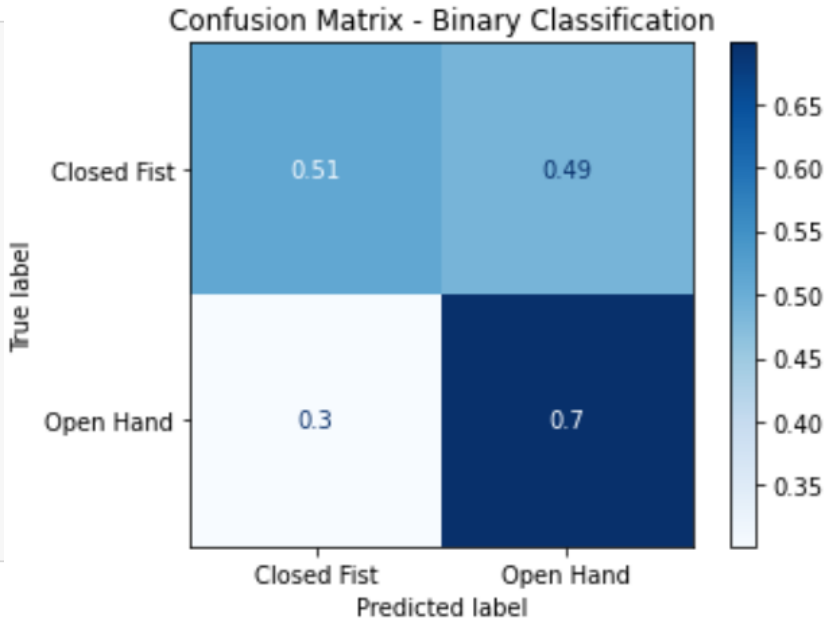
`sklearn.metrics`.recall_score

`sklearn.metrics`.f1_score

© https://www.kdnuggets.com/2020/04/performance-evaluation-metrics-classification.html

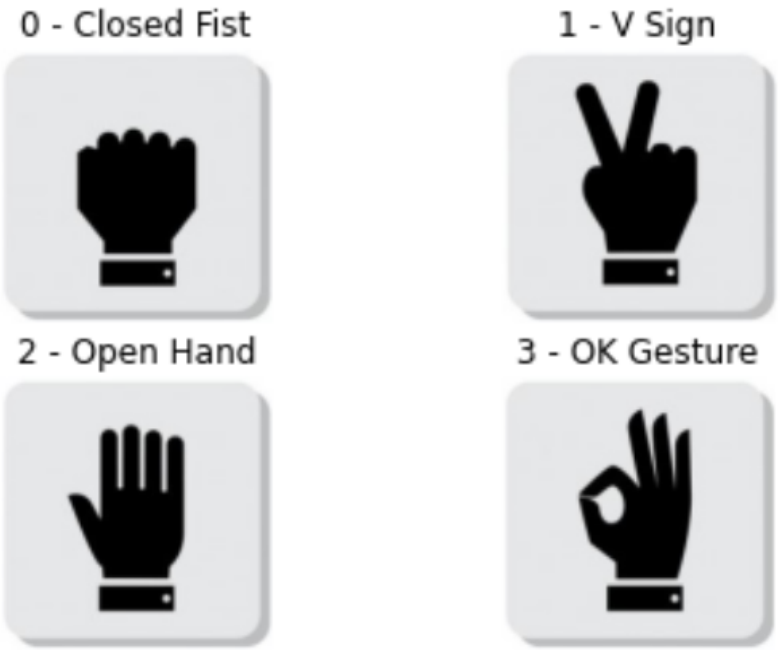
EVALUATING CLASSIFICATION PERFORMANCE

```python
# import the metrics class
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
cnf_matrix = confusion_matrix(y_test_binary, y_pred_binary)
print('Confusion Matrix\n', cnf_matrix)

plt_cnf_matrix = plot_confusion_matrix(logreg, X_test_binary, y_test_binary,
                                       display_labels=['Closed Fist','Open Hand'],
                                       cmap=plt.cm.Blues,
                                       normalize='true')
plt_cnf_matrix.ax_.set_title('Confusion Matrix - Binary Classification')

# Print Evaluation Metrics
print("Accuracy:",accuracy_score(y_test_binary, y_pred_binary))
print("Precision:",precision_score(y_test_binary, y_pred_binary))
print("Recall:",recall_score(y_test_binary, y_pred_binary))
print("F1 Score:",f1_score(y_test_binary, y_pred_binary))
```




```
Confusion Matrix
 [[301 287]
 [176 407]]
Accuracy: 0.6046114432109309
Precision: 0.5864553314121037
Recall: 0.6981132075471698
F1 Score: 0.6374314800313234
```

MULTI-CLASS CLASSIFICATION



`sklearn.metrics`.classification_report

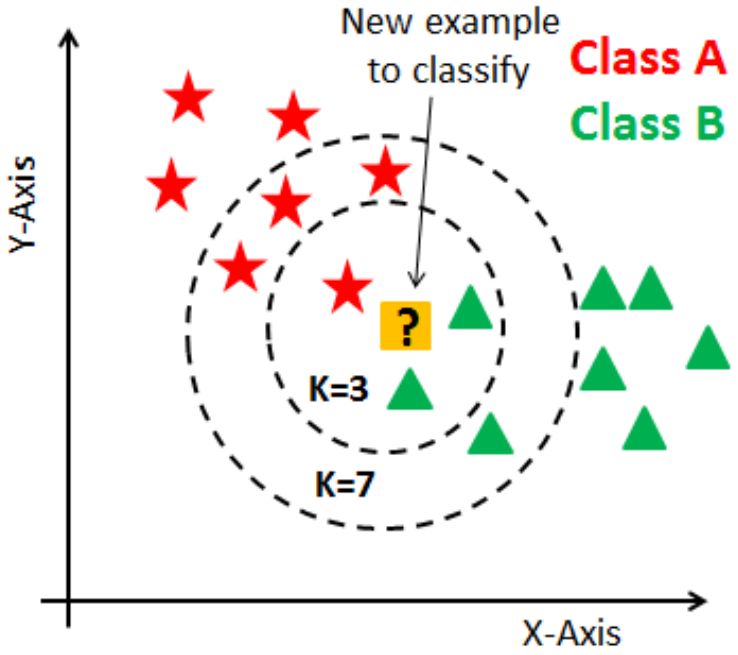
Build a text report showing the main classification metrics for multi-class classification.

K-NEAREST NEIGHBOR

K-Nearest Neighbor, also known as KNN, is a supervised learning algorithm that can be used for regression and classification problems.

KNN is widely used for classification problems in machine learning.

KNN works on a principle assuming every data point falling near each other falls in the same class. That means similar things are near to each other.



https://www.youtube.com/watch?v=9zS3aQGztQo

https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

K-NEAREST NEIGHBOR

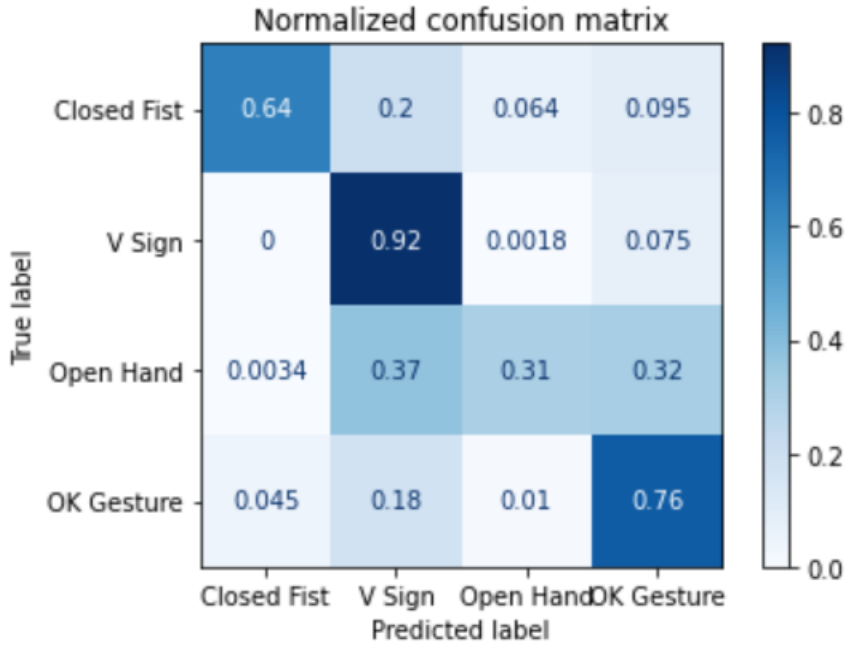
sklearn.neighbors.KNeighborsClassifier

```python
# Import the class
from sklearn.neighbors import KNeighborsClassifier

# Instantiating the model (using the default parameters)
MODEL_KNN = KNeighborsClassifier(n_neighbors=5)

# Train the Model
MODEL_KNN.fit(X_train,y_train)

# Save the Trained Model
pickle.dump(MODEL_KNN, open('Model_KNeighborsClassifier.pkl', 'wb'))

# Predict the Trained Model on our Test data
y_pred_KNN = MODEL_KNN.predict(X_test)

# Print the Classification Report and Confusion Matrix
classifier_performance(MODEL_KNN,y_pred_KNN)
```

K-NEAREST NEIGHBOR

Classification Report:

	precision	recall	f1-score	support
Closed Fist	0.93	0.64	0.76	597
V Sign	0.54	0.92	0.68	571
Open Hand	0.80	0.31	0.44	595
OK Gesture	0.60	0.76	0.67	573
accuracy			0.65	2336
macro avg	0.72	0.66	0.64	2336
weighted avg	0.72	0.65	0.64	2336



The support is the number of occurrences of each class in y_true

DECISION TREE

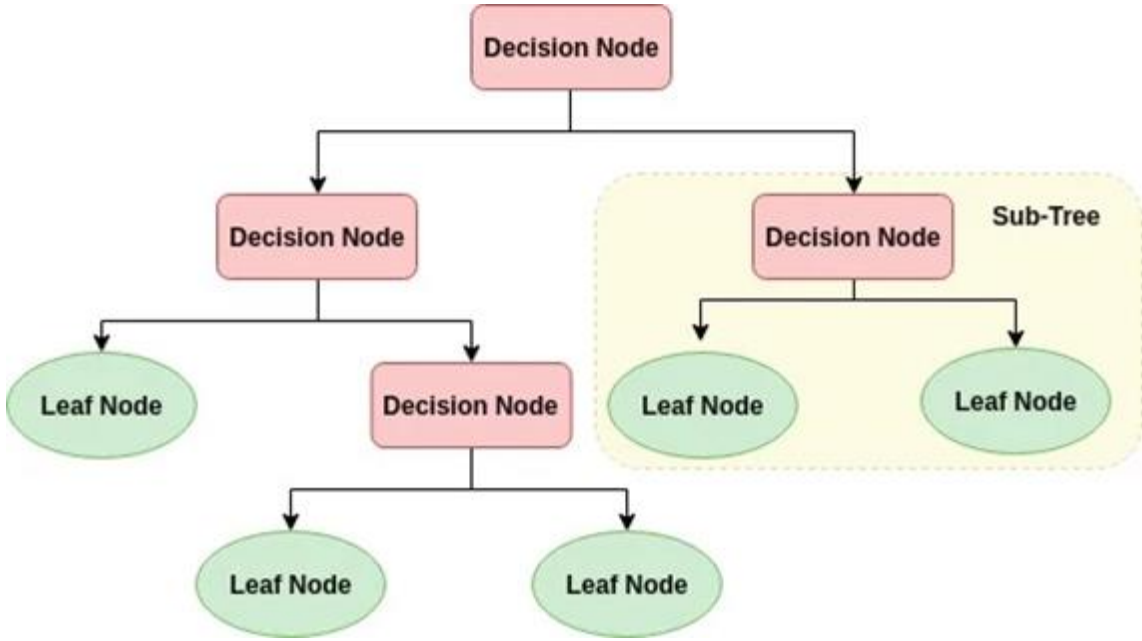
A decision tree builds a classification or regression model in the form of a tree structure.

It utilizes an if-then rule set, which is mutually exclusive and exhaustive for classification.

The rules are learned sequentially using the training data one at a time.

Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition.



https://www.youtube.com/watch?v=JcI5E2Ng6r4

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

DECISION TREE

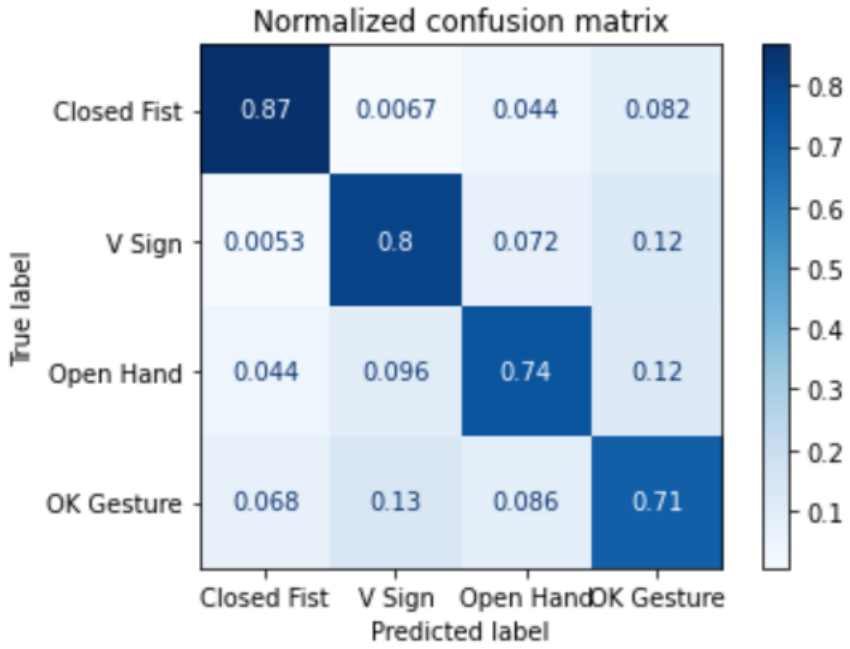
sklearn.tree.DecisionTreeClassifier

```
# Import the class
from sklearn.tree import DecisionTreeClassifier

# Instantiating the model (using the default parameters)
MODEL_DT = DecisionTreeClassifier()

# Train the Model
MODEL_DT.fit(X_train,y_train)

# Save the Trained Model
pickle.dump(MODEL_DT, open('DecisionTreeClassifier.pkl', 'wb'))

# Predict the Trained Model on our Test data
y_pred_DT = MODEL_DT.predict(X_test)

# Print the Classification Report and Confusion Matrix
classifier_performance(MODEL_DT,y_pred_DT)
```

DECISION TREE

Classification Report:

	precision	recall	f1-score	support
Closed Fist	0.88	0.87	0.88	597
V Sign	0.77	0.80	0.78	571
Open Hand	0.79	0.74	0.77	595
OK Gesture	0.69	0.71	0.70	573
accuracy			0.78	2336
macro avg	0.78	0.78	0.78	2336
weighted avg	0.78	0.78	0.78	2336

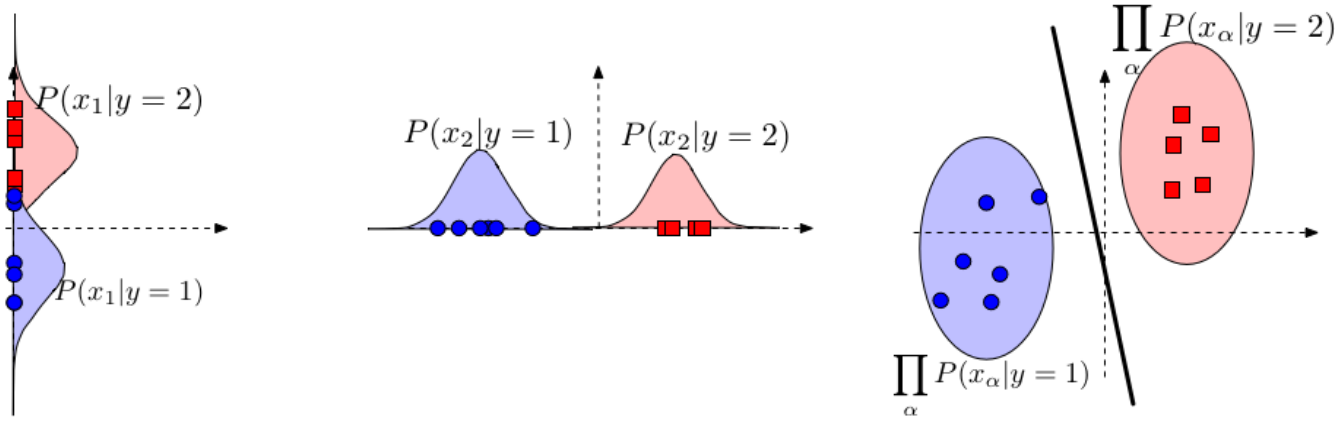


NAIVE BAYES

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem.

- They are fast and easy to implement.
- They require a small amount of training data to estimate the necessary parameters.
- But The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome.

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution (bell shaped curve which is symmetric about the mean of the feature values).



https://www.youtube.com/watch?v=HZGCoVF3YvM&t=627s

https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn

NAIVE BAYES

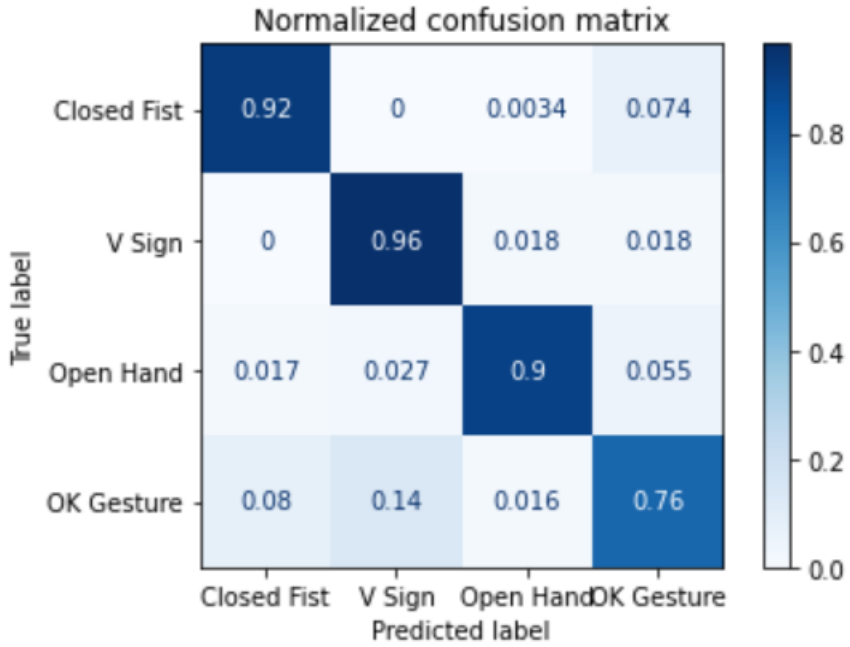
sklearn.naive_bayes.GaussianNB

```
# Import the class
from sklearn.naive_bayes import GaussianNB

# Instantiating the model (using the default parameters)
MODEL_GNB = GaussianNB()

# Train the Model
MODEL_GNB.fit(X_train,y_train)

# Save the Trained Model
pickle.dump(MODEL_GNB, open('Model_GaussianNB.pkl', 'wb'))

# Predict the Trained Model on our Test data
y_pred_GNB = MODEL_GNB.predict(X_test)

# Print the Classification Report and Confusion Matrix
classifier_performance(MODEL_GNB,y_pred_GNB)
```

NAIVE BAYES

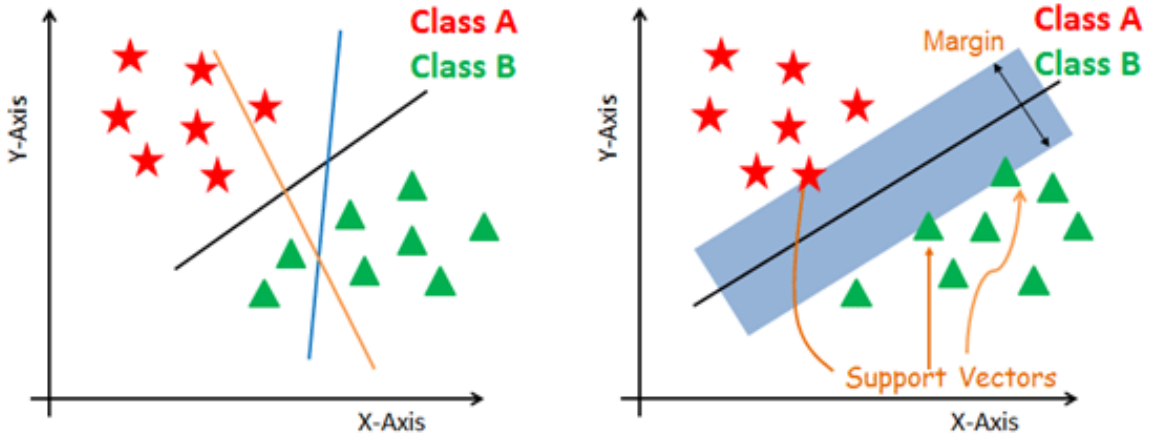
Classification Report:

	precision	recall	f1-score	support
Closed Fist	0.91	0.92	0.92	597
V Sign	0.85	0.96	0.90	571
Open Hand	0.96	0.90	0.93	595
OK Gesture	0.83	0.76	0.80	573
accuracy			0.89	2336
macro avg	0.89	0.89	0.89	2336
weighted avg	0.89	0.89	0.89	2336



SUPPORT VECTOR MACHINES

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks. However, it is mostly used in classification problems.
- The main objective is to segregate the given dataset in the best possible way. The distance between the nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset.



https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

SUPPORT VECTOR MACHINES

sklearn.svm.SVC

```
# Import the class
from sklearn.svm import SVC

# Instantiating the model (using the default parameters)
MODEL_SVM = SVC()

# Train the Model
MODEL_SVM.fit(X_train,y_train)

# Save the Trained Model
pickle.dump(MODEL_SVM, open('Model_SupportVectorClasifier.pkl', 'wb'))

# Predict the Trained Model on our Test data
y_pred_SVM = MODEL_SVM.predict(X_test)

# Print the Classification Report and Confusion Matrix
classifier_performance(MODEL_SVM,y_pred_SVM)
```

SUPPORT VECTOR MACHINES

Classification Report:

	precision	recall	f1-score	support
Closed Fist	0.93	0.93	0.93	597
V Sign	0.88	0.99	0.93	571
Open Hand	0.95	0.86	0.90	595
OK Gesture	0.86	0.85	0.86	573
accuracy			0.90	2336
macro avg	0.91	0.91	0.90	2336
weighted avg	0.91	0.90	0.90	2336