CSAI 801 Project: COVID-19 Outcome Prediction

Adham Mokhtar - 20398545

Project Description:

The data used in this project will help to identify whether a person is going to recover from coronavirus symptoms or not based on some pre-defined standard symptoms. These symptoms are based on guidelines given by the World Health Organization (WHO).

Dataset Description:

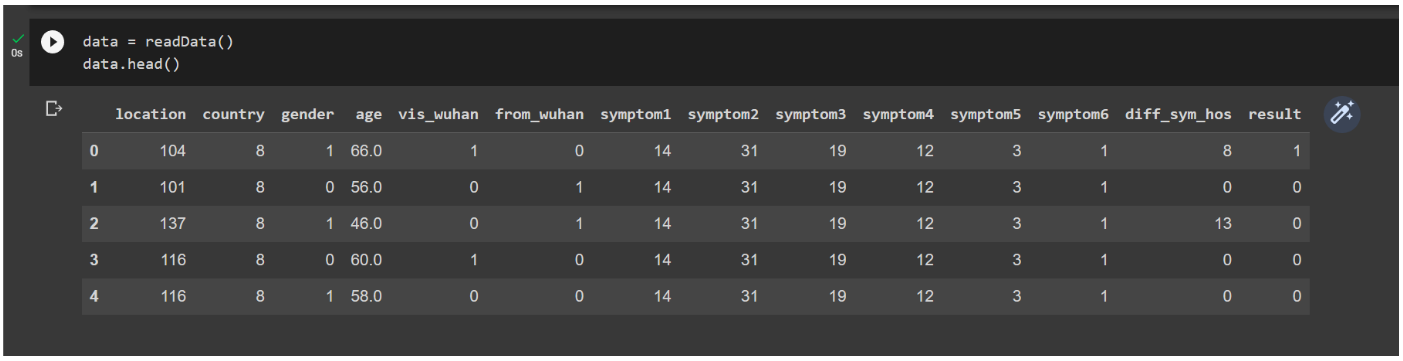
This dataset has daily level information on the number of affected cases, deaths and recovery from 2019 novel coronavirus. Please note that this is a time series data and so the number of cases on any given day is the cumulative number.

The data is available from 22 Jan, 2020. Data is in "data.csv".

The dataset contains 14 major variables that will be having an impact on whether someone has recovered or not, the description of each variable are as follows:

1. Country: where the person resides
2. Location: which part in the Country
3. Age: Classification of the age group for each person, based on WHO Age Group Standard
4. Gender: Male or Female
5. Visited Wuhan: whether the person has visited Wuhan, China or not
6. From Wuhan: whether the person is from Wuhan, China or not
7. Symptoms: there are six families of symptoms that are coded in six fields.

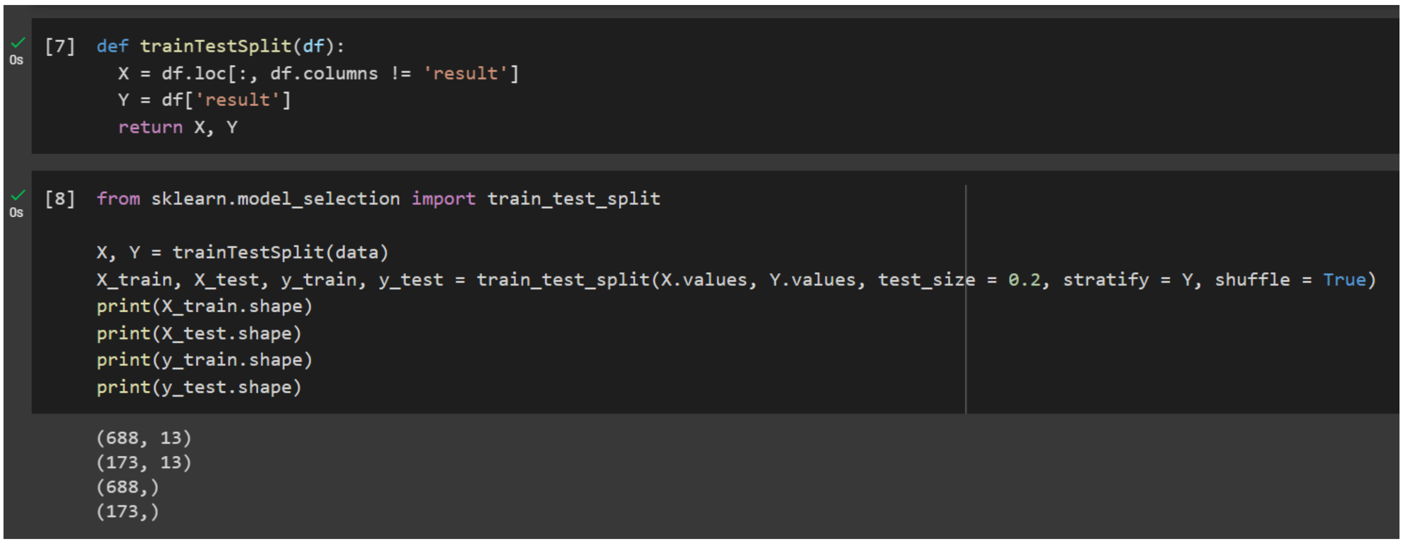
13. Time before symptoms appear
14. Result: death (1) or recovered (0)



```
data = readData()
data.head()
```

	location	country	gender	age	vis_wuhan	from_wuhan	symptom1	symptom2	symptom3	symptom4	symptom5	symptom6	diff_sym_hos	result
0	104	8	1	66.0	1	0	14	31	19	12	3	1	8	1
1	101	8	0	56.0	0	1	14	31	19	12	3	1	0	0
2	137	8	1	46.0	0	1	14	31	19	12	3	1	13	0
3	116	8	0	60.0	1	0	14	31	19	12	3	1	0	0
4	116	8	1	58.0	0	0	14	31	19	12	3	1	0	0

The data supposed to be cleaned and preprocessed, but it has a **negative values** in the *diff_sys_hos* column so we had to remove it before **splitting the data into training and testing**.



```
[7] def trainTestSplit(df):
      X = df.loc[:, df.columns != 'result']
      Y = df['result']
      return X, Y
```

```
[8] from sklearn.model_selection import train_test_split

    X, Y = trainTestSplit(data)
    X_train, X_test, y_train, y_test = train_test_split(X.values, Y.values, test_size = 0.2, stratify = Y, shuffle = True)
    print(X_train.shape)
    print(X_test.shape)
    print(y_train.shape)
    print(y_test.shape)

    (688, 13)
    (173, 13)
    (688,)
    (173,)
```

Classifiers:

As required, I built different classifiers and compared them:

1. K-Nearest Neighbors.
2. Logistic Regression.
3. Naïve Bayes.
4. Decision Trees.
5. Support Vector Machines.

I did these operations and metrics for each classifier (check the code for more details):

- Grid Search
    - Best hyper-parameters
    - Cross-validation test accuracy comparison
- Classification Report
    - Precision
    - Recall
    - F1-score
    - Classes support
    - Accuracy
    - Precision vs Recall Curve
    - ROC/AUC
    - Confusion Matrix

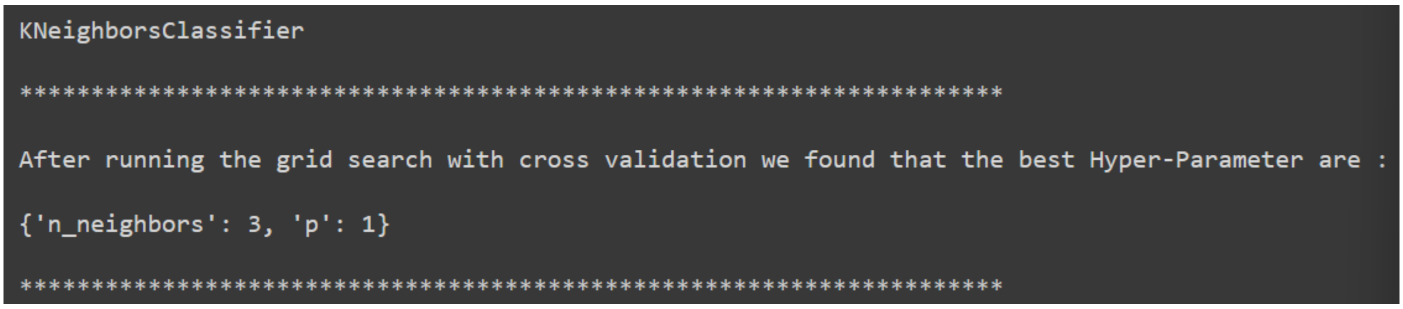
Here **Grid Search** is used to find the best **hyper-parameters** using the **Cross-Validation.**

The best hyper-parameters that lead to the highest accuracy is shown below.

1- Using K-Nearest Neighbors:

After perform the Grid Search on K-nearest neighbors with hyper-parameters:

- n_neighbors: 1 to 19
- p: 1 and 2

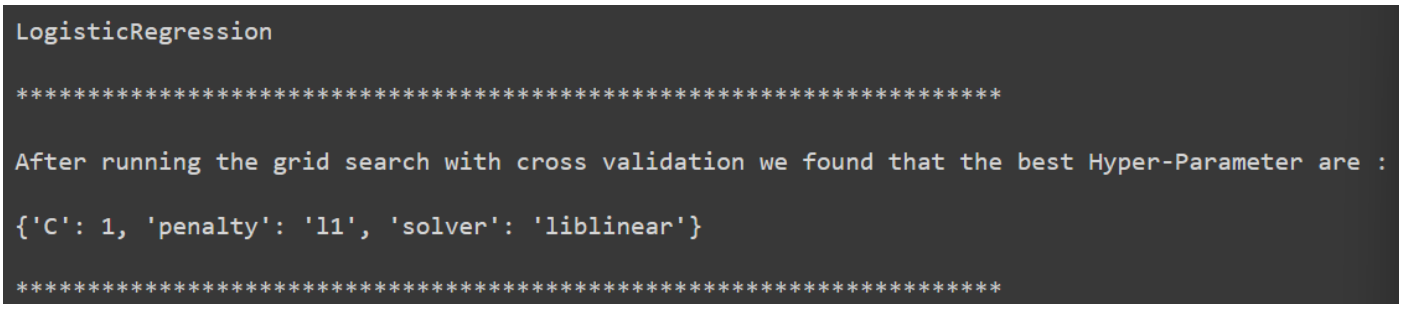


```
KNeighborsClassifier

*********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'n_neighbors': 3, 'p': 1}

*********************************************************************
```

2- Logistic Regression:

After perform the Grid Search on Logistic Regression with hyper-parameters:

- penalty: L1, L2
- solver: liblinear
- C: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3

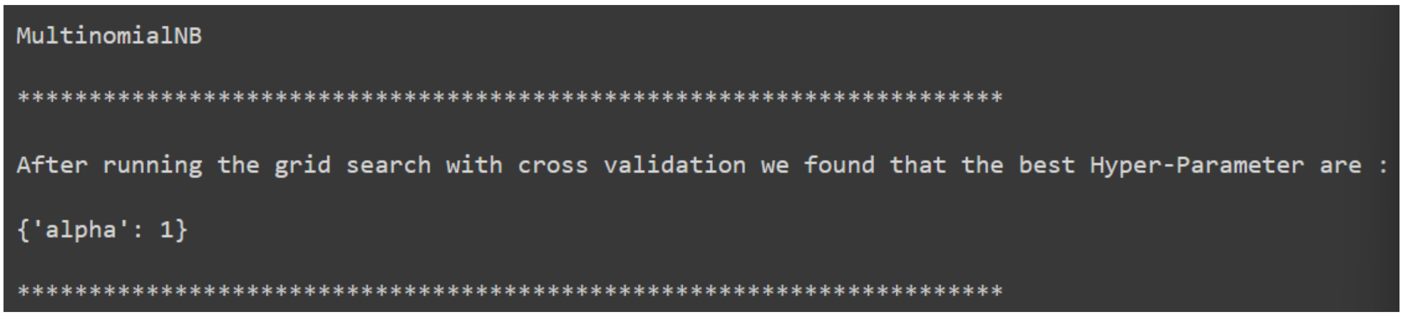


```
LogisticRegression

*********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}

*********************************************************************
```

3- Naïve Bayes

After perform the Grid Search on Naïve Bayes (**Multinomial**) with hyper-parameters:

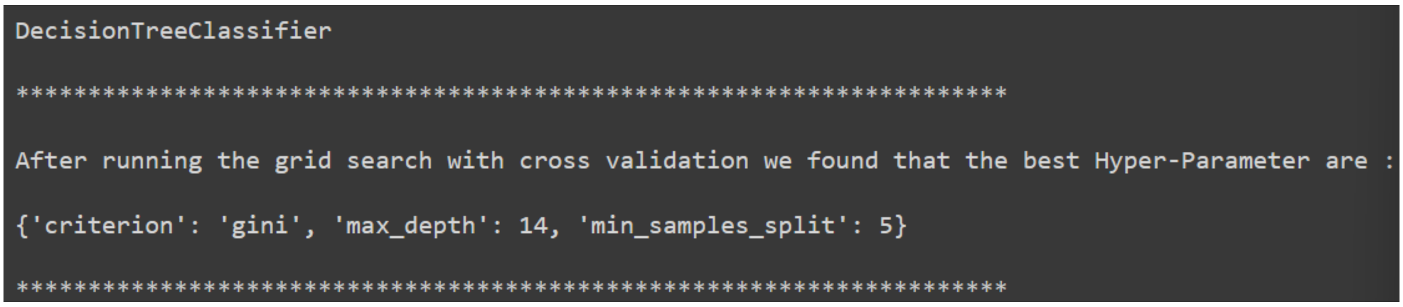
- alpha: 1 to 10


```
MultinomialNB

**********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'alpha': 1}

**********************************************************************
```

4- Decision Trees

After perform the Grid Search on Decision Trees with hyper-parameters:

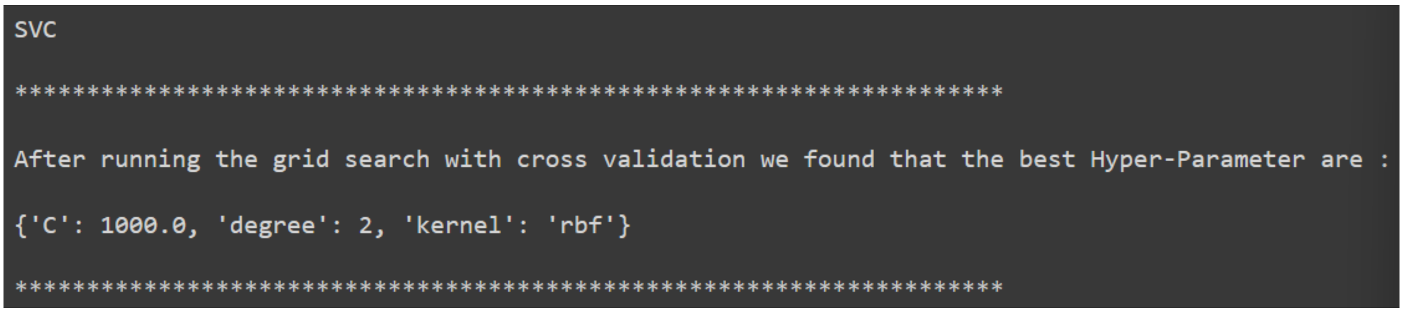
- criterion: gini and entropy
- max_depth: 8 to 14
- min_samples_split: 5, 10 and 15


```
DecisionTreeClassifier

**********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'criterion': 'gini', 'max_depth': 14, 'min_samples_split': 5}

**********************************************************************
```

5- Support Vector Machines

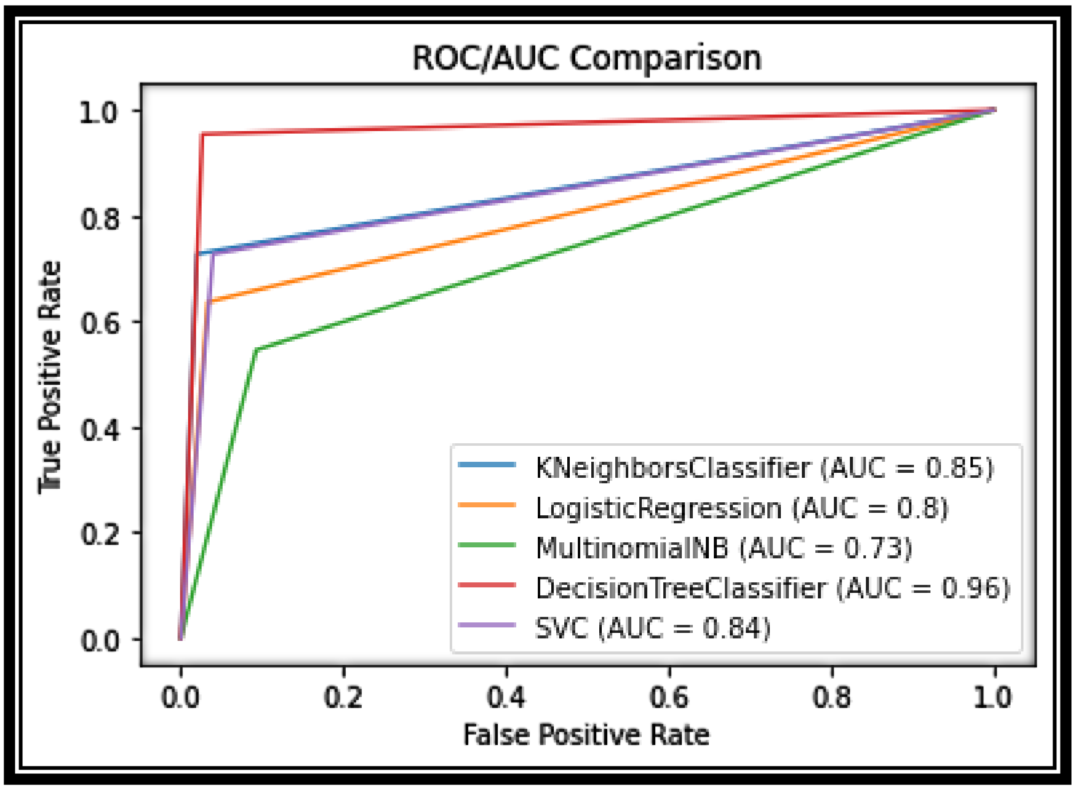
After perform the Grid Search on Support Vector Machines with hyper-parameters:

- kernal: linear, rbf and poly
- degree: 1 to 4
- C: 1e-3, 1e-2, 1, 1e2, 1e3


```
SVC

**********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'C': 1000.0, 'degree': 2, 'kernel': 'rbf'}

**********************************************************************
```

Results:

And if we take the ROC/AUC metrics for comparison:



And from this table which shows the performance among different metrics

Model	Accuracy (%)	F1-score	Precision	Recall
KNN	95	0.78	90	85
Logistic regression	92	0.68	84	80
Multinominal	86	0.50	70	73
Decision Tree	97	0.89	92	96
SVM	93	0.73	84	84

They all show that the **Decision Tree classifier** has the best results.

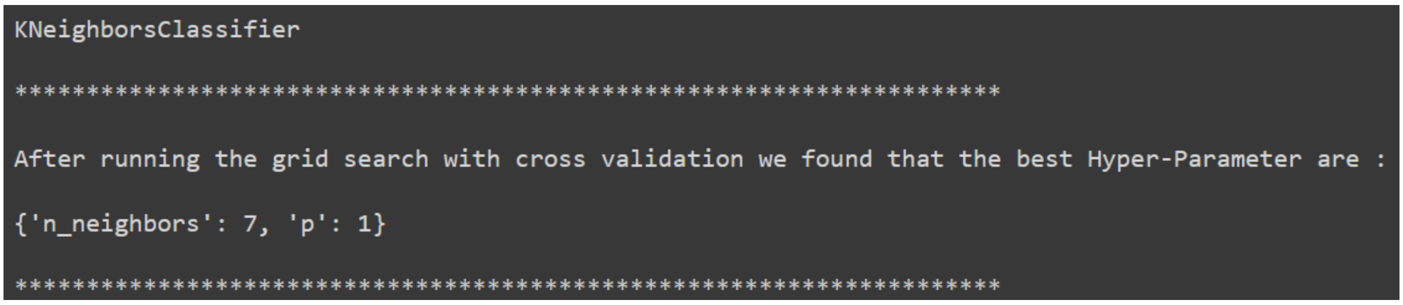
Another Approach:

If we add a preprocessing step and convert categorical features to bins and one hot encoding:

1- Using K-Nearest Neighbors:

After perform the Grid Search on K-nearest neighbors with hyper-parameters:

- n_neighbors: 1 to 19
- p: 1 and 2

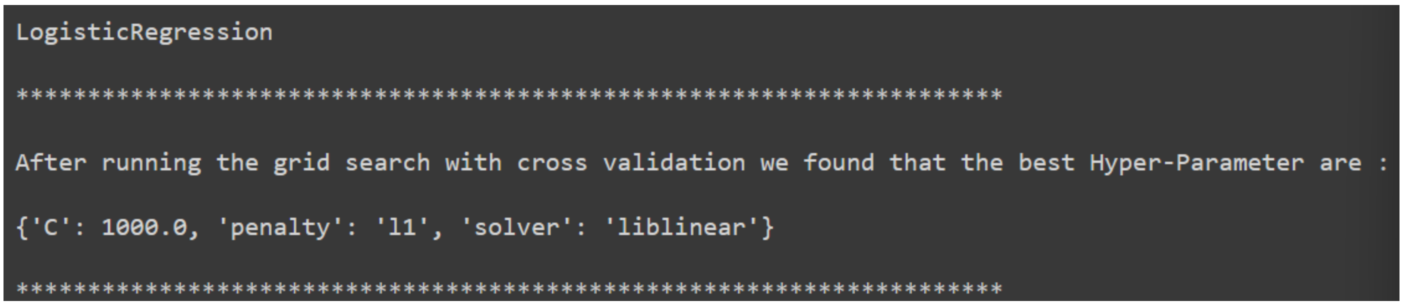


```
KNeighborsClassifier

**********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'n_neighbors': 7, 'p': 1}

**********************************************************************
```

2- Logistic Regression:

After perform the Grid Search on Logistic Regression with hyper-parameters:

- penalty: L1, L2
- solver: liblinear
- C: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3

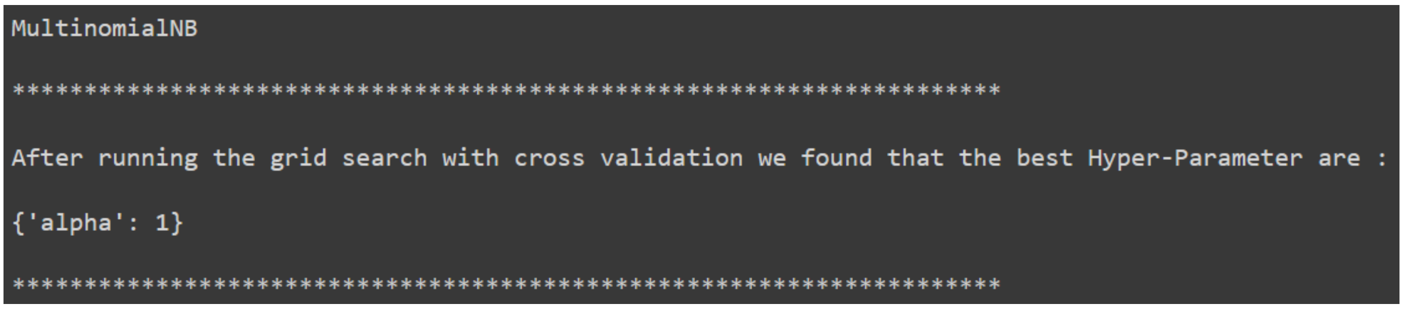


```
LogisticRegression

**********************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'C': 1000.0, 'penalty': 'l1', 'solver': 'liblinear'}

**********************************************************************
```

3- Naïve Bayes

After perform the Grid Search on Naïve Bayes (**Multinomial**) with hyper-parameters:

- alpha: 1 to 10

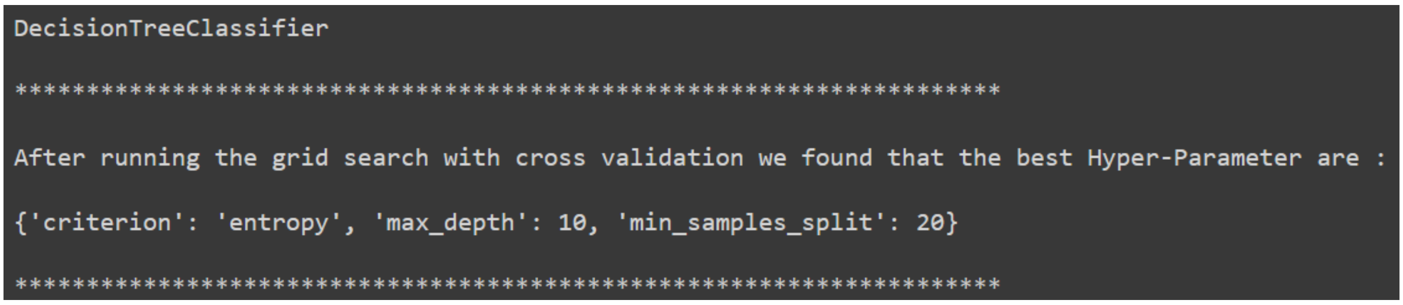


```
MultinomialNB

**************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'alpha': 1}

**************************************************************
```

4- Decision Trees

After perform the Grid Search on Decision Trees with hyper-parameters:

- criterion: gini and entropy
- max_depth: 8 to 14
- min_samples_split: 5, 10 and 15

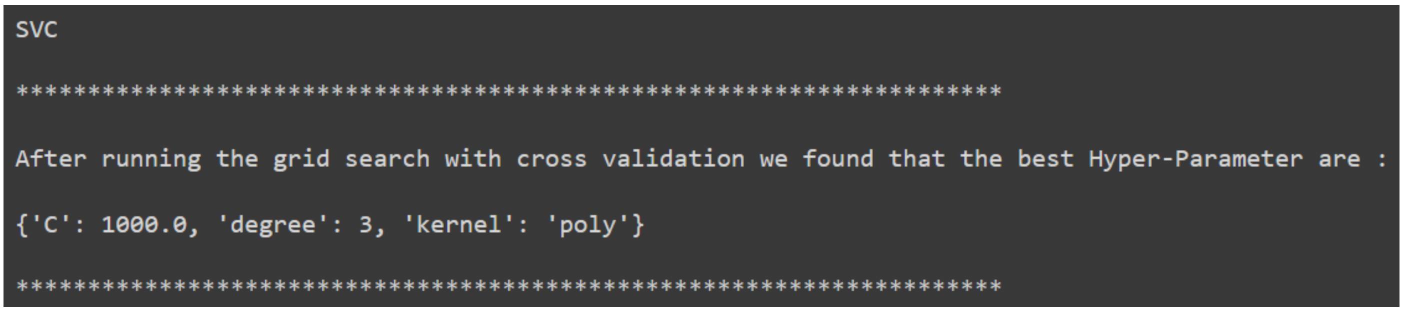


```
DecisionTreeClassifier

**************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 20}

**************************************************************
```

5- Support Vector Machines

After perform the Grid Search on Support Vector Machines with hyper-parameters:

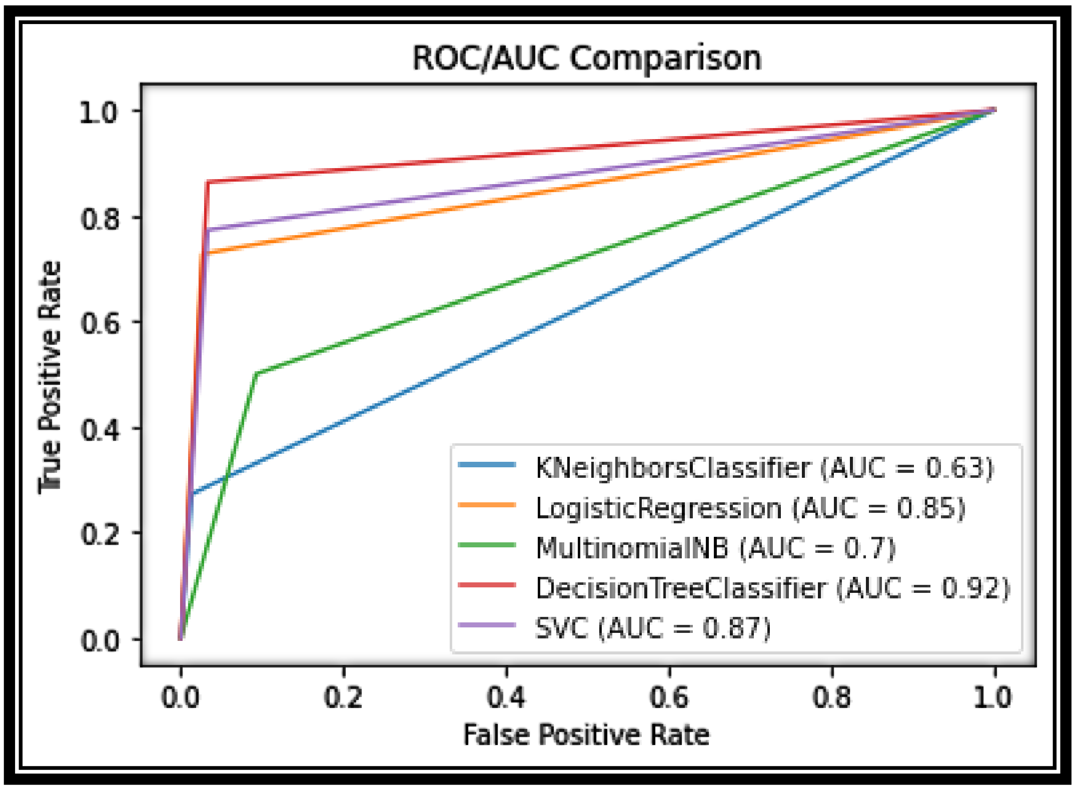
- kernal: linear, rbf and poly
- degree: 1 to 4
- C: 1e-3, 1e-2, 1, 1e2, 1e3



```
SVC

**************************************************************

After running the grid search with cross validation we found that the best Hyper-Parameter are :

{'C': 1000.0, 'degree': 3, 'kernel': 'poly'}

**************************************************************
```

Results:

And if we take the ROC/AUC metrics for comparison:



And from this table which shows the performance among different metrics

Model	Accuracy (%)	F1-score	Precision	Recall
KNN	90	0.40	83	63
Logistic regression	94	0.76	88	85
Multinominal	86	0.47	68	70
Decision Tree	95	0.83	89	92
SVM	94	0.77	87	87

They all also show that the **Decision Tree classifier** has the best results.

For more result, visualization and graphs please check the code.

Thank you.
Adham Mokhtar
20398545