**Table of Content**

[1 Introduction 4](#_Toc37536217)

[1.1 Problem Description and Goal 4](#_Toc37536218)

[1.2 The Important Of The Project 4](#_Toc37536219)

[1.3 Project Goal 4](#_Toc37536220)

[2 Data Exploration 5](#_Toc37536221)

[2.1 About Dataset 5](#_Toc37536222)

[2.2 About Attribute 5](#_Toc37536223)

[2.2.1 Data Types 6](#_Toc37536224)

[2.2.2 Data Balance 6](#_Toc37536225)

[2.2.3 Missing values and outliers 6](#_Toc37536226)

[2.2.4 Distributions of attributes using plots 7](#_Toc37536227)

[2.2.4.1 Count plot 7](#_Toc37536228)

[2.2.4.2 Histogram 9](#_Toc37536229)

[3 Data preprocessing 10](#_Toc37536230)

[3.1 Details of preprocessing data 10](#_Toc37536231)

[3.2 Train/test splits 10](#_Toc37536232)

[3.3 Feature selection 11](#_Toc37536233)

[4 Models 12](#_Toc37536234)

[4.1 Naïve Bayes model 12](#_Toc37536235)

[4.2 Decision Tree model 12](#_Toc37536236)

[4.3 Random Forest model 12](#_Toc37536237)

[4.4 Extra Tree model 12](#_Toc37536238)

[5 Results 13](#_Toc37536239)

[5.1 Performance metrics 13](#_Toc37536240)

[5.1.1 Comparison of models’ performance using plots 13](#_Toc37536241)

[5.1.1.1 Comparing performance Accuracy for each Models 13](#_Toc37536242)

[5.1.1.2 Comparing performance Recall for each Models 14](#_Toc37536243)

[5.1.1.3 Comparing performance Precision for each Models 15](#_Toc37536244)

[5.1.1.4 Comparing performance Confusion Matrix for each Models 16](#_Toc37536245)

[5.1.2 Which model you select and why: 16](#_Toc37536246)

[6 Tools and libraries 17](#_Toc37536247)

[7 Difficulties and challenges 17](#_Toc37536248)

[8 Future work 17](#_Toc37536249)

[9 Work division 17](#_Toc37536250)

[10 References 17](#_Toc37536251)

# Introduction

## Problem Description and Goal

The purpose of this dataset is to identify an English uppercase letter from a black and white image with 20 different fonts and different styles.

## The Important Of The Project

Letter recognition is very important, because it can enable us to scan an image and recognize all the letters and word included whatever the font or style applied to it, and this might be very efficient when wanting to capture a big paragraph and copy it immediately without the need to write this big paragraph.

## Project Goal

The goal of this project is to enable the algorithm to identify the uppercase letter from the image, and the result should be accurate and reliable to meet the expectation and need of the users.

A close up of a logo

Description automatically generated

For the sake of this project we will be dealing with classification task. Classification is the process of finding a model that describes and distinguishes data classes and concepts, and in this dataset, we have 26 class that represent the uppercase alphabetical letter from A to Z.

# Data Exploration

## About Dataset

We have chosen this dataset from the website UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Letter+Recognition>), this dataset represent the values of an uppercase letters from A-Z in black and white images, the letters are in 20 different font and our goal is to try to identify the letter using the attribute values specified in this dataset. The dataset file was a DATA file, so we convert to a csv file to be able to work with it.

This dataset consists of 17 columns and 20,000 rows.

A close up of a logo

Description automatically generated

## About Attribute

As we mentioned above there are 17 attributes, one of them represents the target which is the capital letters and the other 15 represent the statistics of the pixels in the binary. All of the 16 attributes in the dataset file didn’t have a row that represent the header, so it was replaces with the default header the excel apply which is Column and the number concatenated with it, but the description of the attribute was included in a separate file.

The “on pixels” represent the image of a desired character, and it’s an array about 45\*45 pixels.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Name in dataset | Name from the description | Description | Attribute Type |
|  | Column1 | Letter | capital letter | (26 values from A to Z) |
|  | Column2 | x-box | horizontal position of box | Integer |
|  | Column3 | y-box | vertical position of box | Integer |
|  | Column4 | Width | width of box | Integer |
|  | Column5 | High | height of box | Integer |
|  | Column6 | Onpix | total # “on pixels” | Integer |
|  | Column7 | x-bar | mean x of “on pixels” in box | Integer |
|  | Column8 | y-bar | mean y of “on pixels” in box | Integer |
|  | Column9 | x2bar | mean x variance | Integer |
|  | Column10 | y2bar | mean y variance | Integer |
|  | Column11 | xybar | mean x y correlation | Integer |
|  | Column12 | x2ybr | mean of x \* x \* y | Integer |
|  | Column13 | xy2br | mean of x \* y \* y | Integer |
|  | Column14 | x-ege | mean edge count left to right | Integer |
|  | Column15 | xegvy | correlation of x-ege with y | Integer |
|  | Column16 | y-ege | mean edge count bottom to top | Integer |
|  | Column17 | yegvx | correlation of y-ege with x | Integer |

### Data Types

The target attribute which represents the uppercase letters data type is object, the other 16 attributes are all integers (int64), all of the 16 attribute are calculate then scaled to a range from 0-15,

### Data Balance

There are in total 26 classes, we could say that the classes frequency is almost balance because the frequency of each class is almost similar.

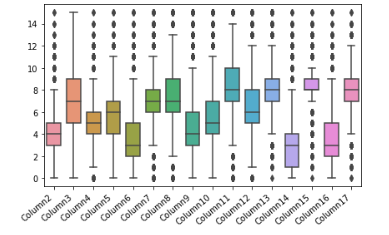
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. |  |  |  |  |  |  |  |  |  |
| Letter |  |  |  |  |  |  |  |  |  |
| Frequency | 789 | 766 | 736 | 805 | 768 | 755 | 773 | 734 | 755 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. |  |  |  |  |  |  |  |  |  |
| Letter |  |  |  |  |  |  |  |  |  |
| Frequency | 747 | 739 | 761 | 792 | 783 | 753 | 803 | 783 | 758 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. |  |  |  |  |  |  |  |  |
| Letter |  |  |  |  |  |  |  |  |
| Frequency | 748 | 796 | 813 | 764 | 752 | 787 | 786 | 734 |

### Missing values and outliers

There are no missing values in the dataset, and as the box plot shown for the attribute there are no outliers because out data as we mentioned before is calculate then scaled to range from 0-15.



### Distributions of attributes using plots

Since all of attributes are numeric attributes ranges from 0-15, we first implemented the count plot to know the frequency of each value, after that, we implemented the histogram for the same purpose.

And for the target which is nominal we also implemented the count plot to visualize the distribution of the classes, and the histogram plot.

#### Count plot

|  |  |
| --- | --- |
| **Target:**  A picture containing drawing  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated |  |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated |  |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated |  |

#### Histogram

**Target**: A picture containing drawing

Description automatically generatedA close up of text on a white background

Description automatically generated

# Data preprocessing

## Details of preprocessing data

First, we Changed the names of the columns by the names provided with the data description (also shown in the tables below) to make it more understandable, instead of just numbering the columns.

|  |  |  |
| --- | --- | --- |
| No. | Name in dataset | Name from the description |
|  | Column1 | Letter |
|  | Column2 | x-box |
|  | Column3 | y-box |
|  | Column4 | Width |
|  | Column5 | High |
|  | Column6 | Onpix |
|  | Column7 | x-bar |
|  | Column8 | y-bar |
|  | Column9 | x2bar |
|  | Column10 | y2bar |
|  | Column11 | Xybar |
|  | Column12 | x2ybr |
|  | Column13 | xy2br |
|  | Column14 | x-ege |
|  | Column15 | Xegvy |
|  | Column16 | y-ege |
|  | Column17 | Yegvx |

and we also checked for the missing value and it shows that there is no missing value, so there is no need to use any techniques to handle the missing values.

## Train/test splits

We divided the dataset into two variables, X contains all the columns (attributes) except the target which is the ‘Capital letter’, Y takes the target only.

We split the dataset 80% for the train and 20% for the test, and stratify the distribution by classes.

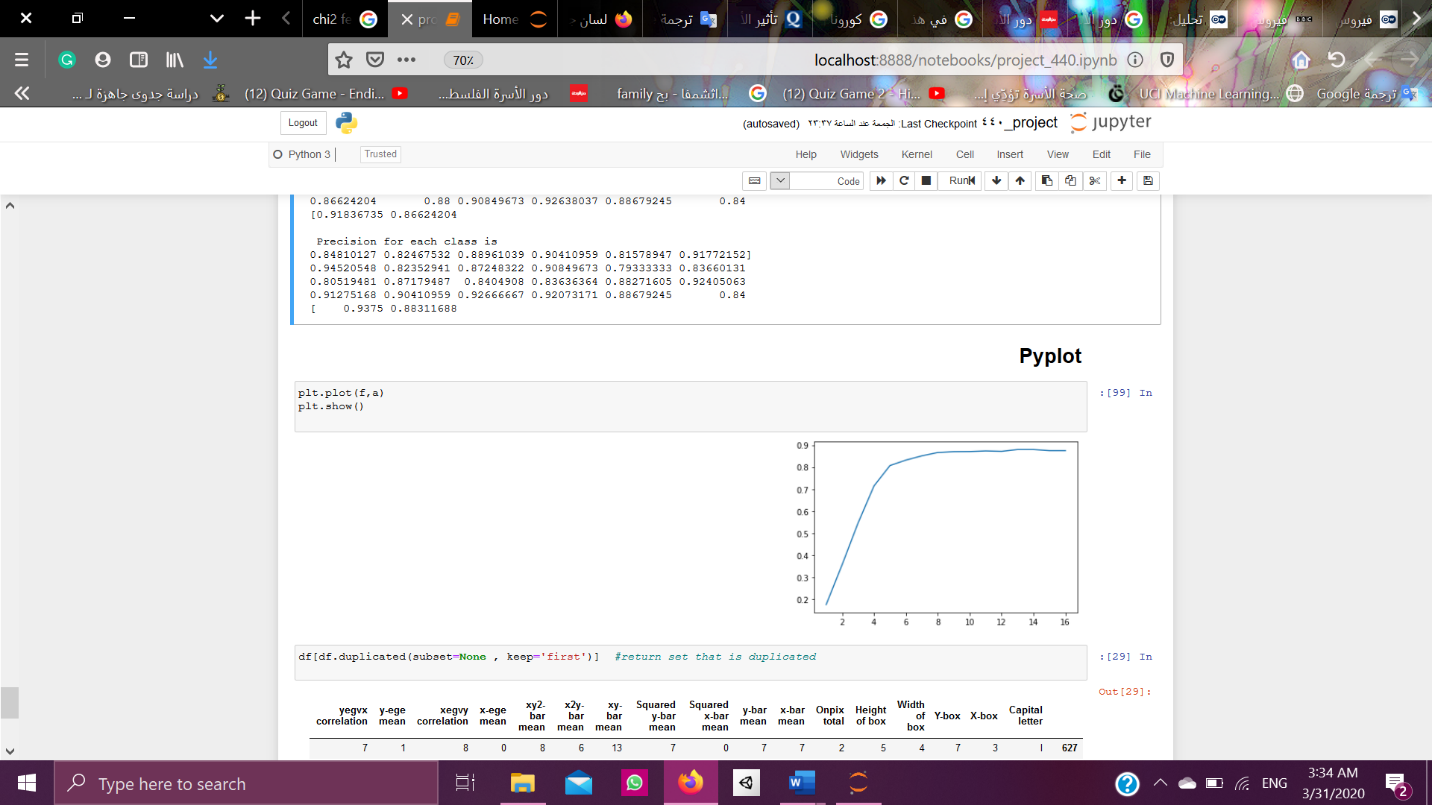
## Feature selection

In the feature selection we use the chi2, and applied some performance matrix such as (accuracy, confusion matrix, recall, precision). And we have achieved the best accuracy using the Decision Tree algorithm with 13 and 14 as the output shows.

A screenshot of a computer

Description automatically generated

All the result (accuracy) can be present it in Pyplot.



# Models

we selected four models to implement in our project: Naive Bayes, Decision Tree, Random Forest and Extra Tree model. we assign 10 as number of cross validation -because we don't have in our paper-.

## Naïve Bayes model

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. we use it because it is commonly used in data mining, and easy to implement. we implement multinomial Naive Bayes classifier because It assumes that the features are drawn from a simple Multinomial distribution. we select alpha and fit\_prior as parameters tuning. alpha parameter it is smoothing parameter and we assign it a default value 1 when train is 80 and test is 20. fit\_prior, it is Boolean parameter and we assign it a default value true.

## Decision Tree model

a decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. we used it because it is commonly used in data mining, and the learning and classification steps of a decision tree are simple and fast. we select two parameters tuning, random state which is the seed used by the random number generator, and criterion the function to measure the quality of a split we assign entropy value for it because it achieves better accuracy than Gini.

## Random Forest model

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We used it because it raised the level of accuracy. we select two parameters tuning, n\_estimators which is the number of trees in the forest. and random state which is controls both the randomness of the bootstrapping of the samples used when building trees.

## Extra Tree model

This class implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy. two parameters tuning

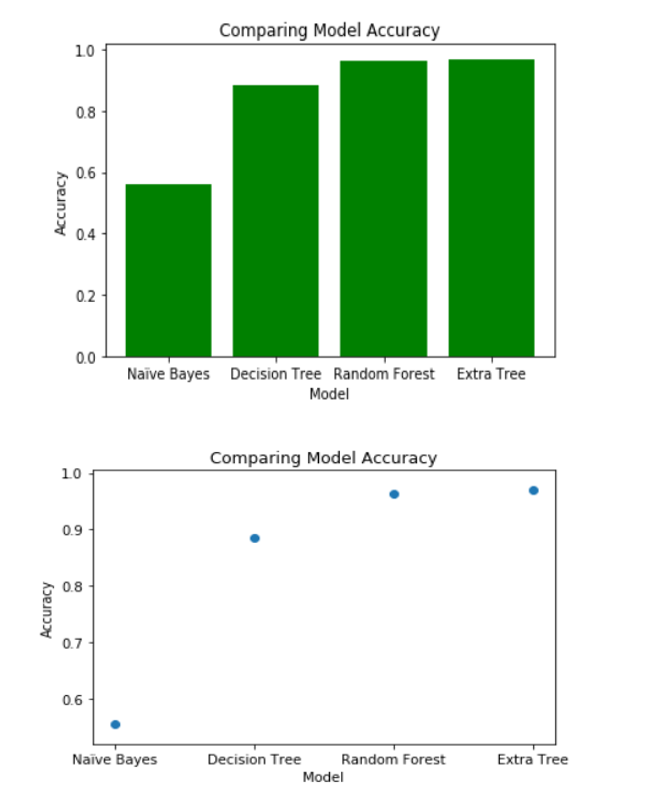
# Results

## Performance metrics

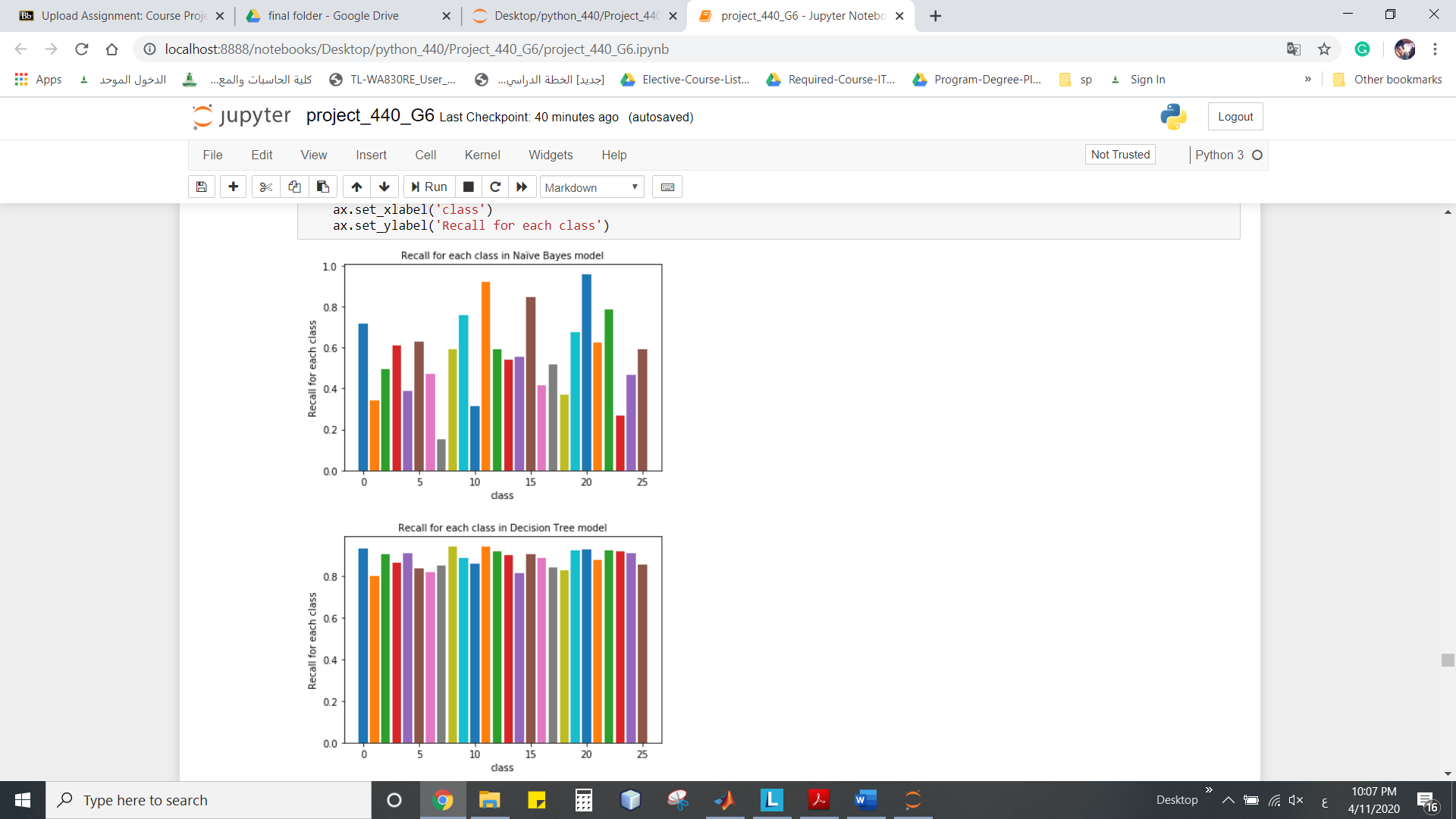
We use accuracy, confusion matrix, recall and precision for each model.

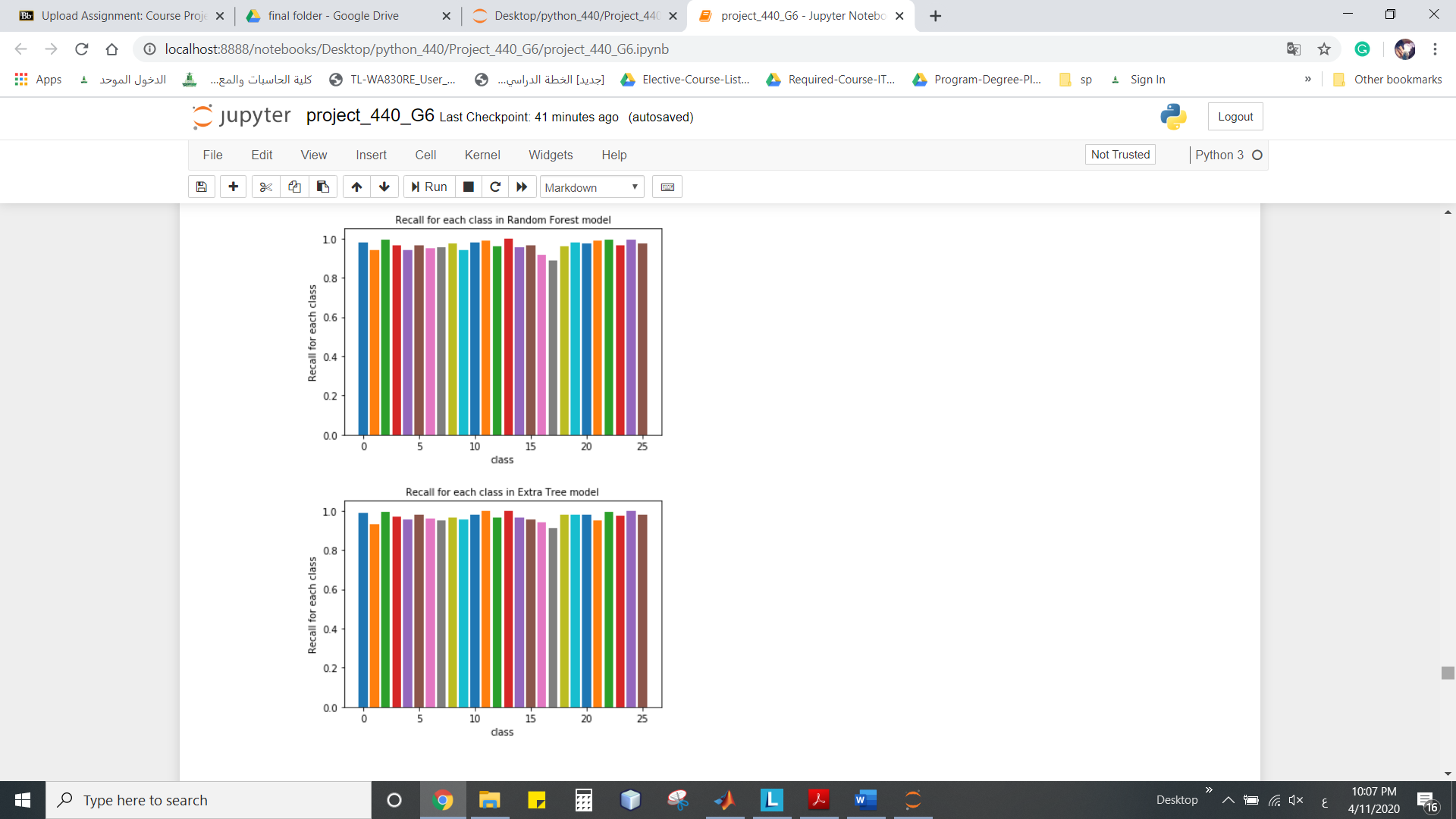
### Comparison of models’ performance using plots

#### Comparing performance Accuracy for each Models

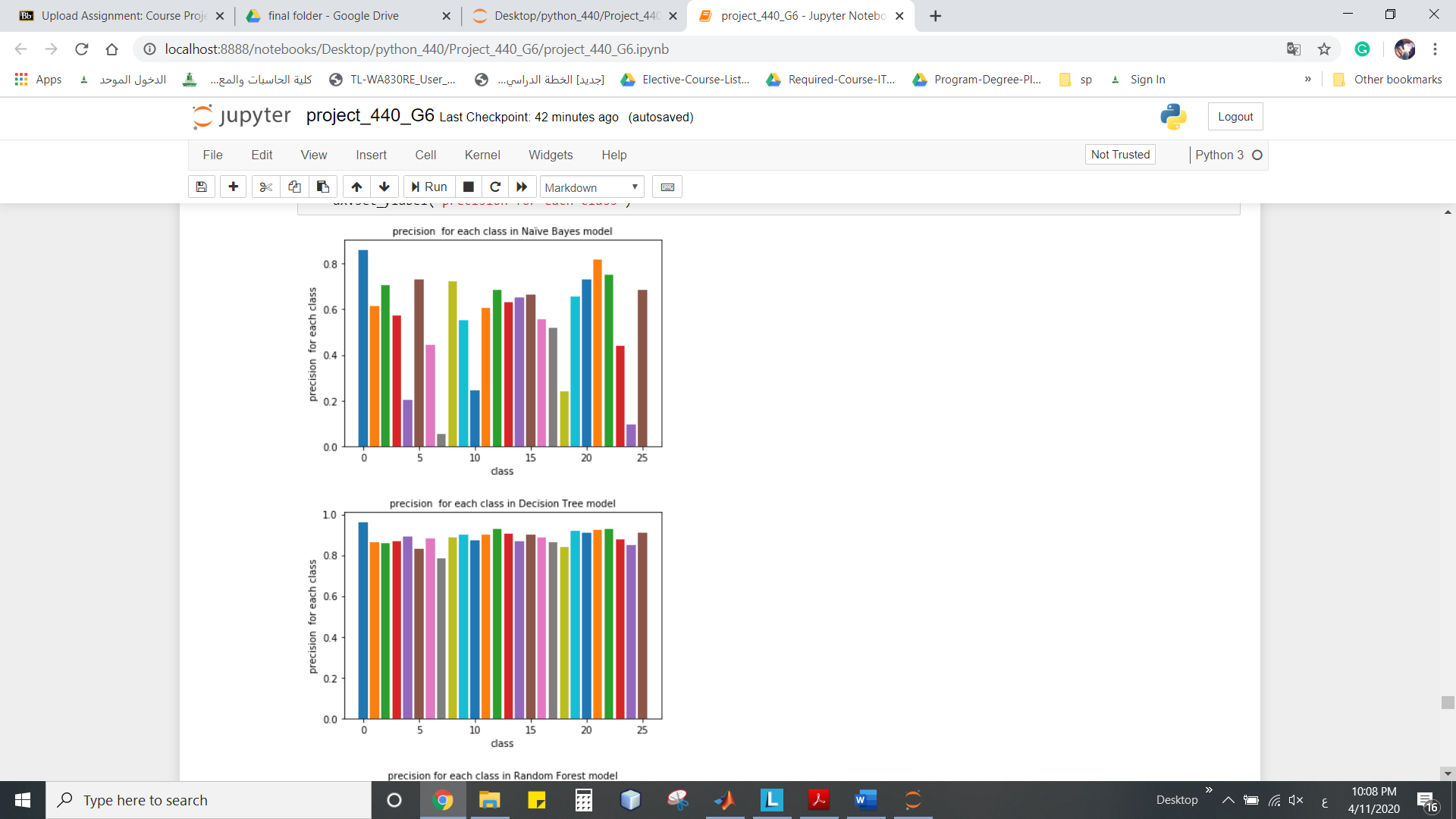


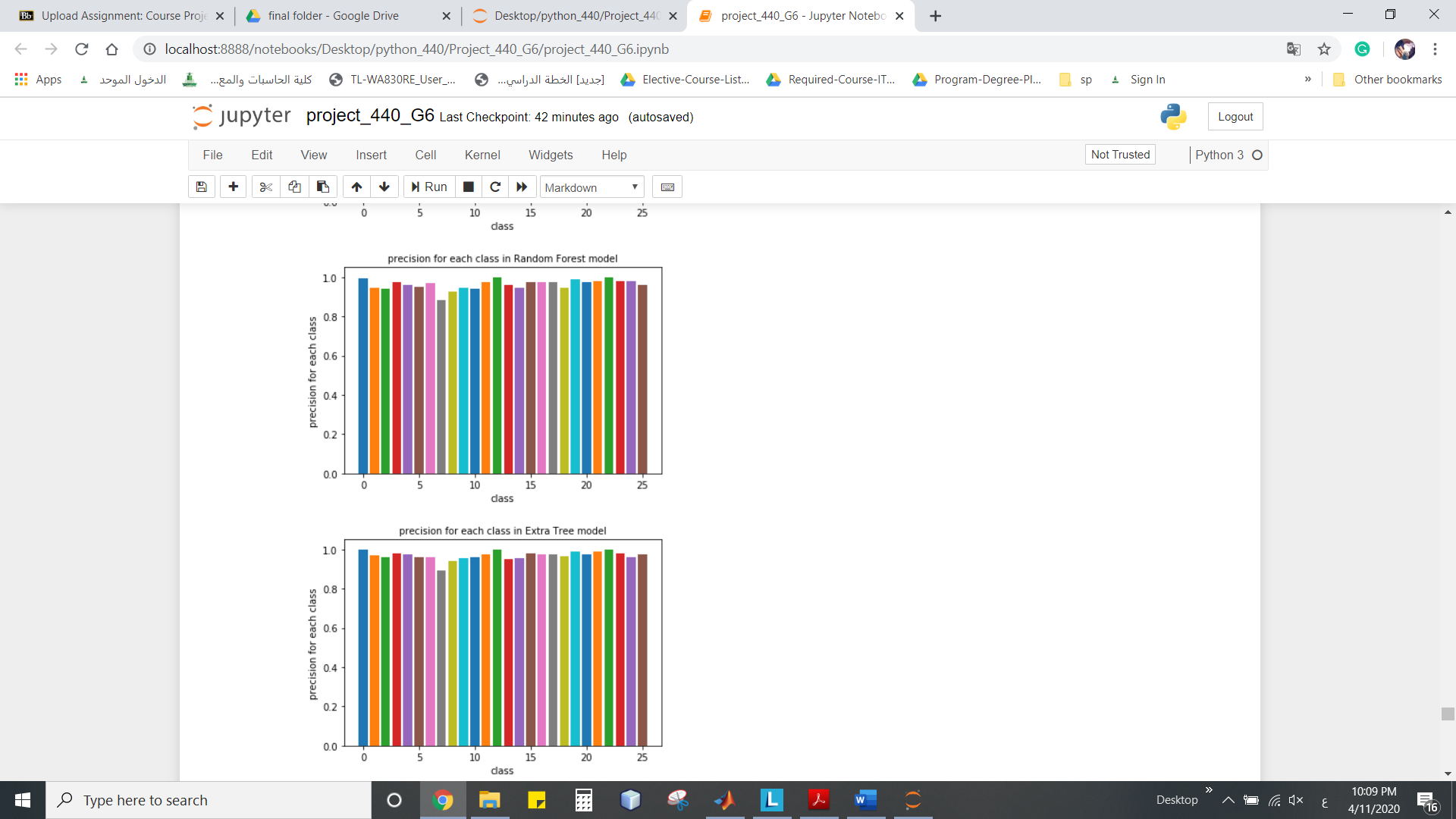
#### Comparing performance Recall for each Models



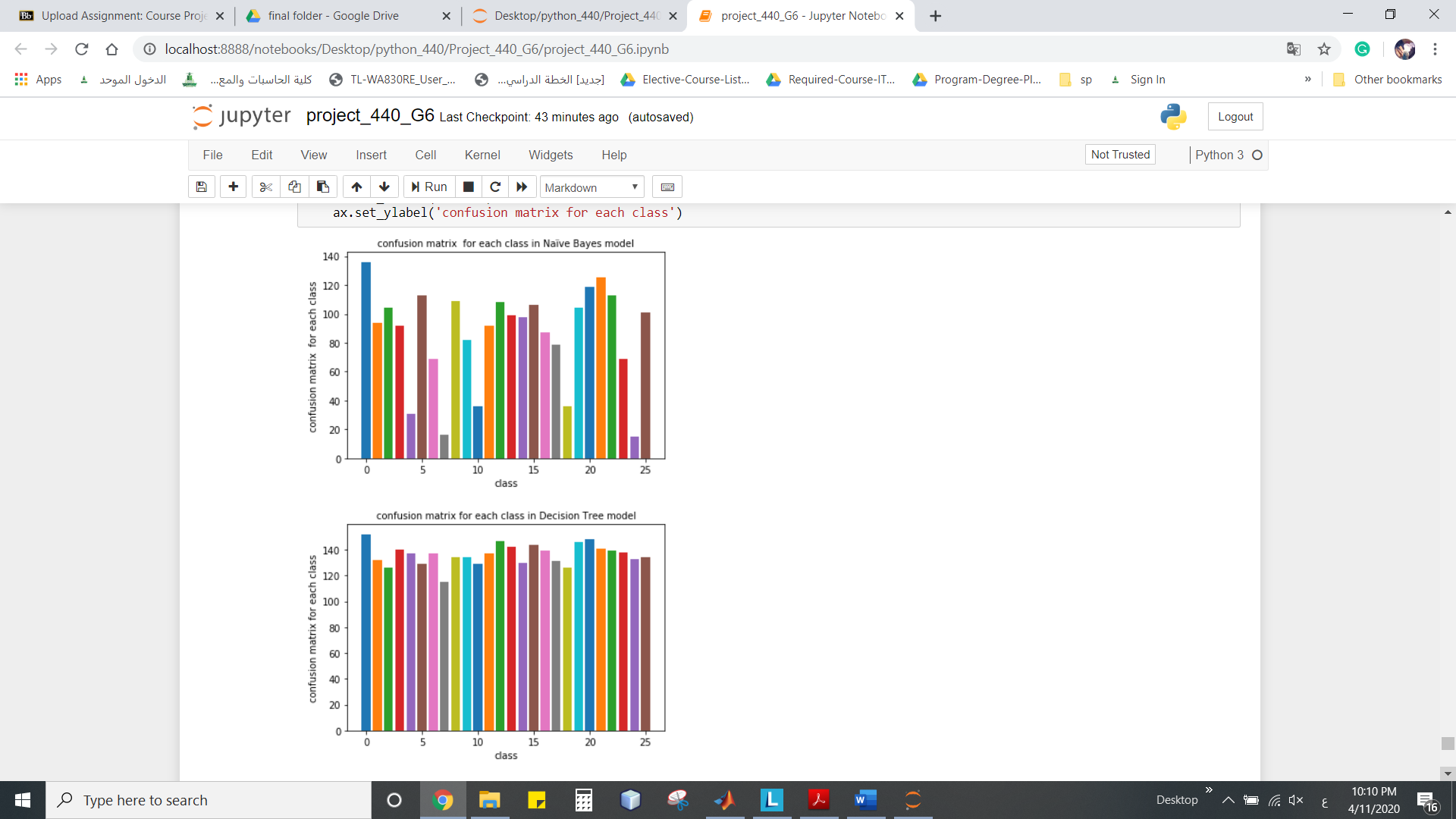


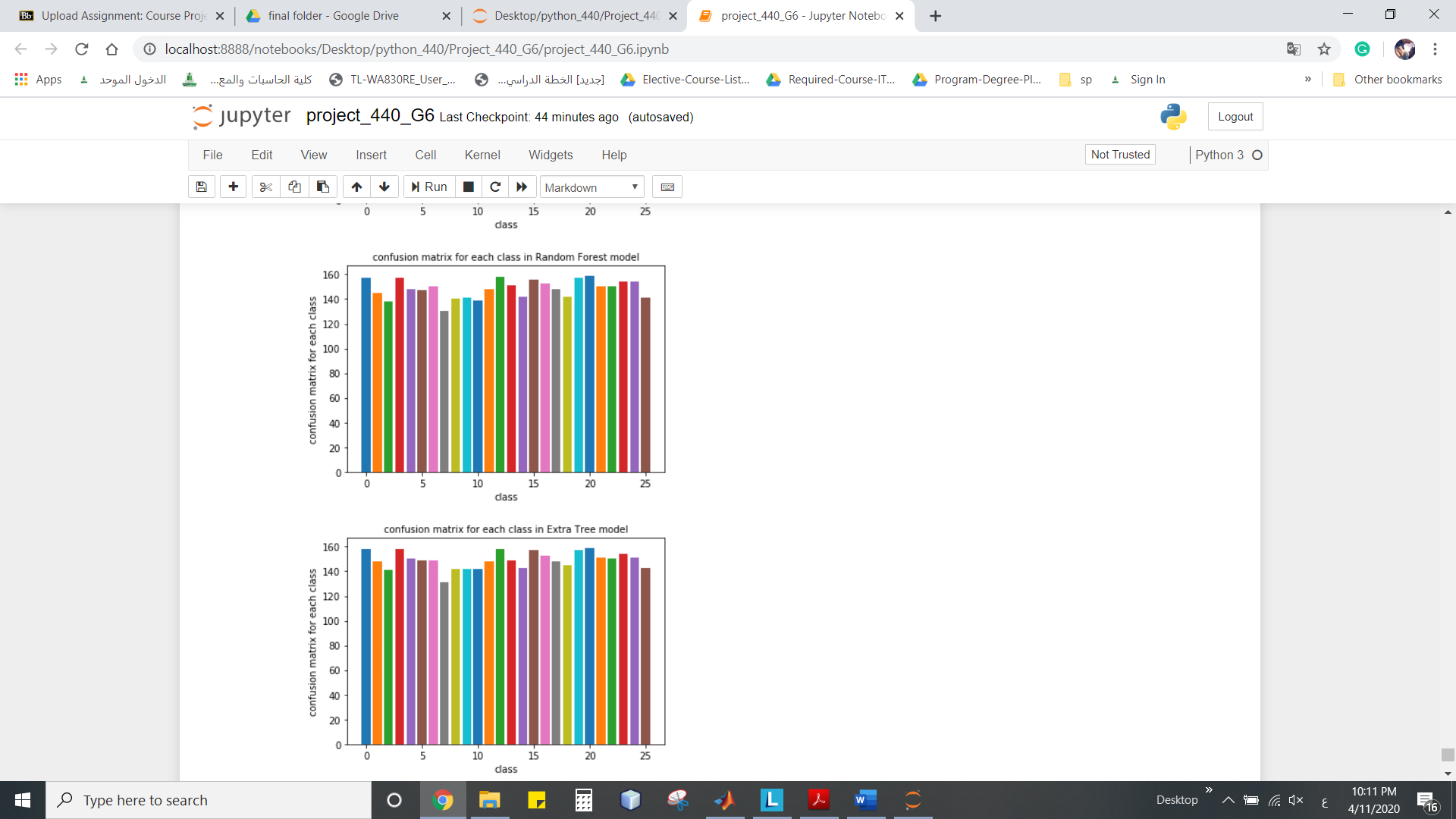
#### Comparing performance Precision for each Models





#### Comparing performance Confusion Matrix for each Models





### Which model you select and why:

Extra Tree model because have the highest values of accuracy 0.973 = 97.3%.

# Tools and libraries

* Python
* Jupiter notebook (anaconda)
* Libraries: pandas, seaborn, matplotlib.pyplot, runpy, matplotlib, sklearn.ensemble, sklearn.tree sklearn.model\_selection, sklearn.naive\_bayes and sklearn.feature\_selection.

# Difficulties and challenges

We had faced some difficulties in coding as we used a new programming language, which is python.

# Future work

Apply our models in a new build Arabic dataset to compare the accuracy.

# References

* <https://archive.ics.uci.edu/ml/datasets/Letter+Recognition>
* <https://link.springer.com/article/10.1007/BF00114162#auth-1>
* <https://scikit-learn.org/stable/> <https://link.springer.com/chapter/10.1007/11941439_114>
* <https://economictimes.indiatimes.com/definition/decision-tree-model>
* <https://stackoverflow.com/questions/42528921/how-to-prevent-overlapping-x-axis-labels-in-sns-countplot>
* https://matplotlib.org/tutorials/introductory/pyplot.html