الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Applications Project

EMAIL SPAM DETECTION

Project Report

Supervisor: Dr. Maisaa Abu Qassem **Submitted By: Owais Hilal 4200051**

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Applications Project

Table Of Content

Applications Project

1.Summary

The project aims to develop a machine learning-based solution for detecting email spam. By leveraging supervised learning techniques, specifically K-Nearest Neighbors (KNN), Decision Trees and Naive Bayse the system learns from labeled email data to classify incoming emails as either spam or non-spam (ham).

Key Components:

- Data Collection: Obtain a dataset containing labeled email data, where each email is labeled as spam or non spam (ham).

- Data Preprocessing: Clean and preprocess the email data, which may involve tasks such as dealing with missing and null values, as well as removing unnecessary columns.

- Model Training: Train machine learning models, including K-Nearest Neighbors(KNN), Decision Trees and Naive Bayes on the labeled email data to learn patterns and relationships between features and target labels.

- Model Evaluation: Evaluate the performance of the trained models using metrics such as accuracy, precision, recall, and F1-score to assess their effectiveness in accurately classifying spam and non-spam (ham) emails.

2.Problem Statement

Email spam is a pervasive problem in modern communication, with millions of users receiving unsolicited and unwanted messages every day. Traditional methods of filtering spam, such as keyword-based filtering and blacklisting, are often ineffective due to the dynamic and evolving nature of spam messages. Therefore, there is a need to develop more robust and accurate spam detection systems that can adapt to changing spam patterns.

One potential solution is to use machine learning techniques, such as Decision tree classifiers, to analyze large datasets of emails and identify patterns that can be used to distinguish between legitimate emails and spam. However, there are several challenges that need to be addressed in developing such a system, including the high dimensionality of email data, the presence of noisy and irrelevant features, and the need for real-time detection.

The aim of this machine learning project is to develop an effective email spam detection system using Decision tree classifiers. The project will involve collecting and preprocessing a large dataset of emails, selecting relevant features, and training a Decision tree model to classify emails as either spam or Ham. The ultimate goal is to develop a system that can accurately and efficiently detect spam emails, with the potential to be used in real-world applications.

3.Machine Learning Techniques

- **Supervised Learning:**

  Supervised learning algorithms learn from labeled data, where each example consists of input features and an associated target label or output. The goal is to learn a mapping from inputs and outputs.

  - **Regression:**

    Regression algorithms are used when the target variable (output) is continuous or numerical. The goal is to predict a quantity rather than a class label.

  - **Classification:**

    Classification algorithms are used when the target variable (output) is categorical or discrete. The goal is to predict a class label of a new instance.

- **Unsupervised Learning:**

  Unsupervised learning algorithms learn from unlabeled data, where there are no target labels provided. The goal is to discover hidden patterns, structures or relationships within the data.

  - **Clustering:**

    Clustering algorithms partition the data into groups or clusters based on similarity of instances. The goal is to group similar instances together while keeping dissimilar instances apart.

  - **Association:**

    Association algorithms discover interesting relationships or associations between variables in large datasets. They are commonly used in market basket analysis, where the goal is to find patterns in transaction data.

Applications Project

- **Machine Learning Techniques Used In This Project:**

  - **Decision Tree:**

    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. The topmost node in the tree is the root node.

  - **K NEAREST NEIGHBORS (KNN):**

    K-Nearest Neighbors (KNN) is a supervised learning algorithm used for classification and regression tasks. It is a non-parametric algorithm, which means it doesn't make any assumptions about the underlying distribution of the data.

    The algorithm works by finding the K nearest data points in the training set to a given input data point, based on some distance metric (such as Euclidean distance). It then predicts the class (in the case of classification) or the value (in the case of regression) of the input data point based on the majority class or the average value of its K nearest neighbors.

- **Naive Bayes:**

  Naive Bayes is a probabilistic algorithm used for classification and is based on Bayes' theorem. It is called "naive" because it assumes that the features used to classify the data are independent of each other, which may not always be true.

  The algorithm calculates the probability of a data point belonging to each class based on the occurrence of its features. It then selects the class with the highest probability as the predicted class for that data point.

  Naive Bayes is commonly used in text classification tasks, such as spam filtering and sentiment analysis. It works well with large datasets and is computationally efficient. However, its performance may suffer when the independence assumption is violated or when the training data is imbalanced or contains a large number of irrelevant features.

4.Dataset Description

The dataset I'm using is on "Kaggle.com", uploaded by "ASHUTOSH_KUN" 2 years ago and it's 30MB size. Here are more informations about the dataset:

- Number of rows and columns: (5271 , 3002).
- Vector size of each email: 5172.
- Missing or Null values: 0.
- Duplicate rows: 0.
- Duplicate columns: 0.
- Datatype of records: int32(3001), object(1).
- Memory usage: 118.5+MB.

- **Attributes Description:**

  The first column indicates the Email number. The last column has the labels for prediction: 1 for spam, and 0 for not spam. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding non-alphabetical characters/words.

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Applications Project

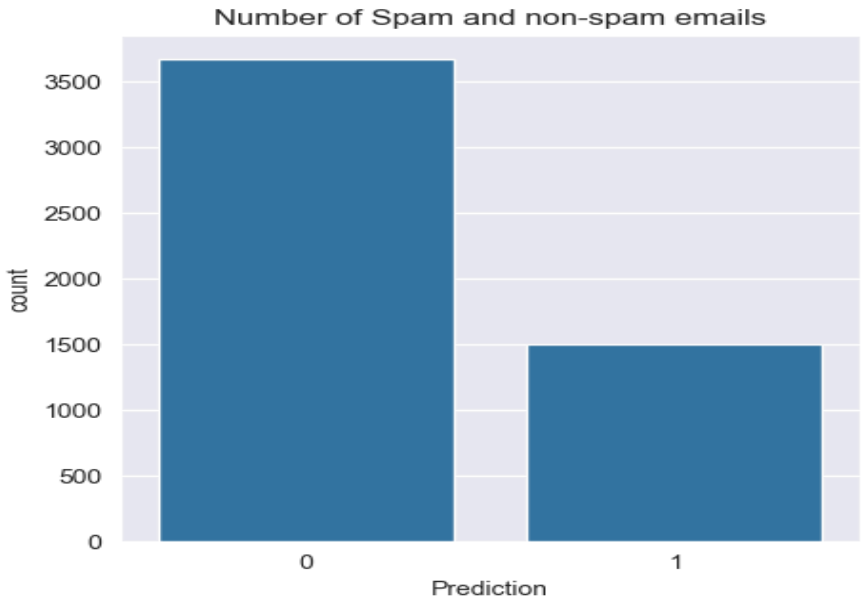
- **The First Few Rows of The Dataset:**

```
email.head()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	0

5 rows × 3002 columns

- **Number of Spam and Non-spam Emails Plot:**



5.Data Preprocessing

Preprocessing data is an important step for data analysis. The following are some benefits of preprocessing data:

- It improves accuracy and reliability. Preprocessing data removes missing or inconsistent data values resulting from human or computer error, which can improve the accuracy and quality of a dataset, making it more reliable.

- It makes data consistent. When collecting data, it's possible to have data duplicates, and discarding them during preprocessing can ensure the data values for analysis are consistent, which helps produce accurate results.

- It increases the data's algorithm readability. Preprocessing enhances the data's quality and makes it easier for machine learning algorithms to read, use, and interpret it.

Applications Project

- **Removing The Stopwords:**

In natural language processing (NLP), stopwords are common words that often appear frequently in text but typically do not carry significant meaning or convey important information about the content. Examples of stopwords include articles (e.g., "the", "a", "an"), prepositions (e.g., "in", "on", "at"), and conjunctions (e.g., "and", "but", "or").

The process of removing stopwords from text data is a fundamental step in many NLP tasks, such as text classification, sentiment analysis, and topic modeling. By eliminating stopwords, we aim to filter out noise and focus on the essential words that contribute to the meaning and context of the text.

We used the “nltk.corpus” library and imported “stopwords” from it, then we defined the following function and applied it to the dataset.

```
nltk.download('stopwords')

def remove_stopwords(df):
    stop_words = set(stopwords.words('english'))
    cols_to_remove = [col for col in df.columns if col.lower() in stop_words]
    df = df.drop(columns=cols_to_remove)

    return df
```

Applications Project

- **The Dataset After Applying The Stopwords Function:**

Applying the stopwords function

```
emails = remove_stopwords(email)
```

The dataset after removing the stopwords

```
emails.head()
```

	Email No.	ect	hou	enron	com	gas	deal	meter	hpl	please	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Email 2	24	27	1	3	1	0	0	0	2	...	0	0	0	0	0	0	0	1	0	0
2	Email 3	1	0	0	0	2	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Email 4	22	10	0	0	0	2	1	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Email 5	17	9	0	0	2	0	3	0	1	...	0	0	0	0	0	0	0	1	0	0

5 rows × 2868 columns

The function reduced the column number from 3002 to 2868, because it removed the stopwords which are unnecessary noisy data.

Applications Project

- **Feature Selection:**

  What is Feature selection? What features are important for your problem statement?
  Choosing the key features for the model is known as feature selection. A feature is a trait that affects or helps solve an issue. While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning.

  In this dataset, the first column that is “Email No.” is of no use. It is just the count of the emails. So if we do not remove this column then we shall end up with some unwanted result.

  Finally, we applied a feature selection function that selects 1400 columns instead of 3001, the function uses “f_classif” criteria as a hyperparameter. Obviously, this function improved my model performance.

Applications Project

- **What is “f_classif”?**

  The "f_classif" (ANOVA F-value) scoring function is commonly used in feature selection for classification tasks. It computes the ANOVA F-value statistic for each feature by comparing the variance between classes with the variance within classes. And here is how it works:

  The dataset is divided into separate groups or classes based on the target variable. Each feature's values are grouped according to the corresponding class labels. For each feature, the mean value is computed for each class. This represents the average value of the feature within each class. The total variance of each feature across all samples is calculated. This represents the variability of the feature regardless of class membership. The variance of each feature within each class is calculated. This represents the variability of the feature within each class. The variance of each feature between classes is calculated. This represents the variability of the feature between different class labels. The F-value statistic is computed for each feature using the formula:

$$F = \frac{between-group\ variance}{within-group\ variance}$$

Applications Project

This ratio quantifies the difference in means between groups relative to the variation within groups. Higher F-values indicate greater discriminatory power of the feature between classes.Finally, based on the computed F-values (and optionally, p-values), the SelectKBest algorithm selects the k features with the highest F-values as the most discriminative features for the classification task.

- **The Dataset After Applying The Feature Selection Function:**

Applying The Feature Selection Function

```
new_x=featureSelect_dataframe(x,y,f_classif,1400)
new_x
```

	ect	hou	enron	com	gas	deal	meter	hpl	please	e	...	fall	fear	hate	debt	reform	plain	valued	lay	military	ff
0	1	0	0	0	0	0	0	0	0	4	...	0	0	0	0	0	0	0	0	0	0
1	24	27	1	3	1	0	0	0	2	141	...	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	2	0	0	0	0	3	...	0	0	0	0	0	0	0	0	0	0
3	22	10	0	0	0	2	1	0	0	79	...	0	0	0	0	0	0	0	0	0	0
4	17	9	0	0	2	0	3	0	1	71	...	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5167	2	0	0	0	0	0	0	0	0	29	...	0	0	0	0	0	0	0	0	0	0
5168	11	3	1	3	5	0	0	0	1	218	...	0	0	0	0	0	0	0	0	0	1
5169	1	0	0	0	0	0	0	0	0	12	...	0	0	0	0	0	0	0	0	0	0
5170	1	0	0	4	0	1	0	0	1	45	...	0	0	0	0	0	0	0	0	0	1
5171	5	2	1	1	5	0	0	0	1	191	...	0	0	0	0	0	0	0	0	0	0

5172 rows × 1400 columns

- **Train-Test Split:**

  The train-test split is a technique for evaluating the performance of a machine learning algorithm.

  It can be used for classification or regression problems and can be used for any supervised learning algorithm.

  The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

  We used the “train_test_split” function from the “sklearn.model_selection” library, the parameters of this function are “new_x” the selected features, “y” the label, with 0.3 test size and random state with value of 42.

Applications Project

- **Dataset Scaling:**

  In machine learning, feature scaling is a preprocessing technique used to standardize or normalize the range of independent features or variables in the dataset. It aims to bring all features to a similar scale to prevent features with larger magnitudes from dominating those with smaller magnitudes. Feature scaling is essential for many machine learning algorithms that are sensitive to the scale of input features, such as gradient descent-based optimization algorithms, k-nearest neighbors (KNN), support vector machines (SVM), and neural networks.

  There are two common methods of feature scaling:

  - Standardization (Z-score normalization): In standardization, each feature is rescaled so that it has a mean of 0 and a standard deviation of 1. This transformation ensures that the feature values are centered around the mean, with a standard deviation that accounts for the variability of the data.

  - Normalization (Min-Max scaling): In normalization, each feature is scaled to a fixed range, usually between 0 and 1. This transformation preserves the relative relationships between feature values and is particularly useful when the features have different units or scales.

6.Building Models

Why do we need to train and test different models?

We need to train and test different models in data mining because it allows us to evaluate the performance of the models and select the one that best suits our needs.

When we train a model, we are teaching it how to make predictions based on the data we provide. We use a portion of the available data (called the training set) to train the model, and we adjust its parameters until it can make accurate predictions.

However, the ultimate goal of building a model is to use it to make predictions on new, unseen data. Therefore, it's important to test the model on a separate set of data (called the testing set) to see how well it performs on new data. This is known as model evaluation.

By testing different models on the same testing set, we can compare their performance and choose the one that performs best. This helps us to avoid overfitting (where the model is too closely tailored to the training data and performs poorly on new data) and to ensure that our model is robust and reliable.

Applications Project

- **Model1 (KNN):**

  The K-Nearest Neighbors (KNN) model is a machine learning algorithm used for both classification and regression tasks. It is a type of instance-based or lazy learning algorithm, which means that it does not create a model during the training phase. Instead, it stores the entire training dataset and uses it to make predictions on new, unseen data.

  The first step of building the model is finding the best hyperparameters according to the dataset we have. So, we used the “GridSearchCV” function from the “sklearn.model_selection” library.

  After applying the function, the output stated the the best hyperparameters are “n_naighbors = 3”, “weights =’distance’ ” and “p = 2”.

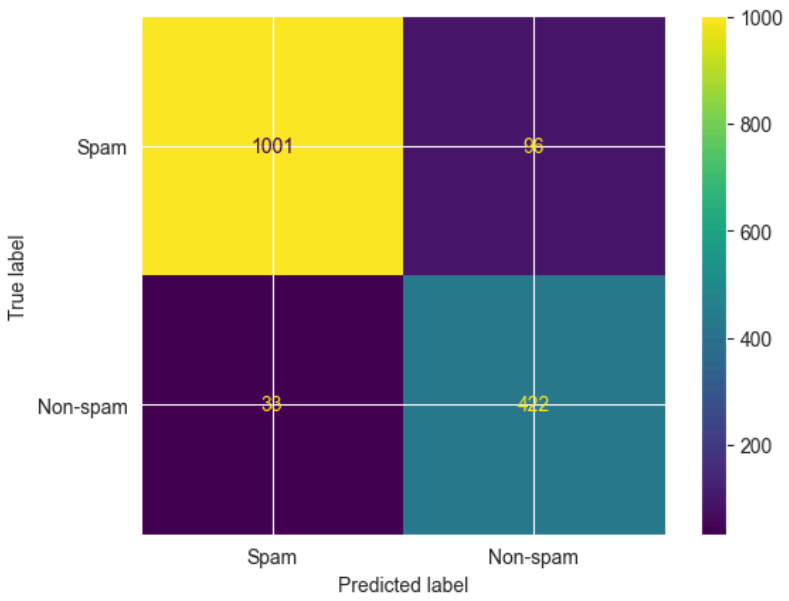
  Now, after we found out what are the best hyperparameters, we built the model with the previously mentioned hyperparameters using the “fit” function. Then we printed the Precision, the Recall, the F1_Score, the Accuracy and the Confusion Matrix.

Applications Project

Here are the results we got:

```
Precision :  0.8146718146718147

Recall :  0.9274725274725275

Accuracy Score :  0.916881443298969

F1 Score :  0.8674203494347379

 [[1001   96]
 [  33  422]]
```

The Confusion Matrix Plot:



- **Model2 (Decision Tree):**

  A decision tree model is a predictive model used in data mining. It uses a tree-like structure to represent a sequence of decisions and their possible outcomes. The tree is built using a dataset of labeled examples, and the algorithm selects the best attribute to split the dataset at each node. Once the tree is constructed, it can be used to classify or predict new examples by following the sequence of decisions. Decision tree models are easy to interpret and visualize, but they can overfit the training data.

  As we did with the previous model, we had to find the best hyperparameters for this model as well. So, we used the same function.

  After applying the function, the output stated that the best hyperparameters are “criterion=’entropy’ ”, “max_depth=30”, “max_features=1000” and “min_sampels_split=10”.

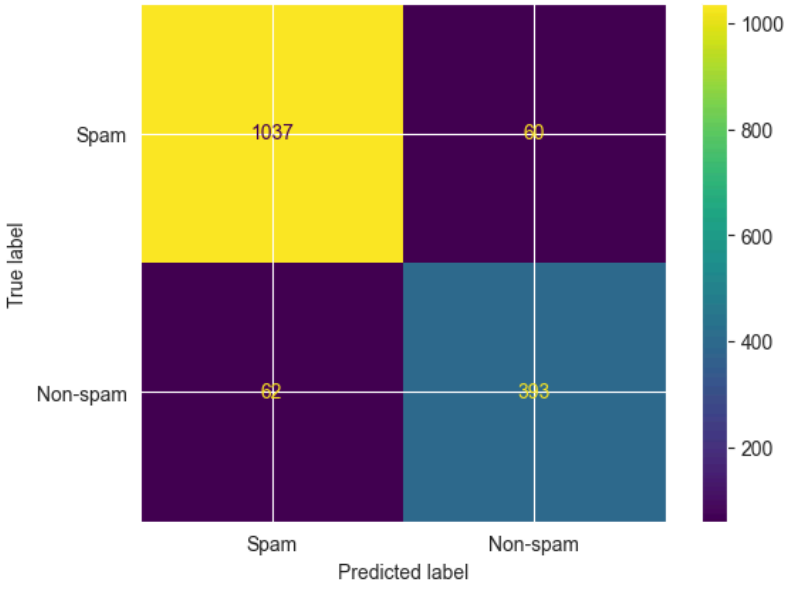
  Now, as we did before. We built the model with the previously mentioned hyperparameters, and printed the results.

Applications Project

Here are the results we got:

```
Precision :  0.8675496688741722

Recall :  0.8637362637362638

Accuracy Score :  0.9213917525773195

F1 Score :  0.8656387665198237

 [[1037   60]
 [  62  393]]
```

The Confusion Matrix Plot:



- **Model3 (Naive Bayse):**

  The Naive Bayes model is a probabilistic machine learning algorithm used for classification tasks in data mining. It is based on Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event.

  The Naive Bayes model assumes that the presence or absence of a feature is independent of the presence or absence of any other feature, given the class variable. This assumption simplifies the calculation of probabilities.

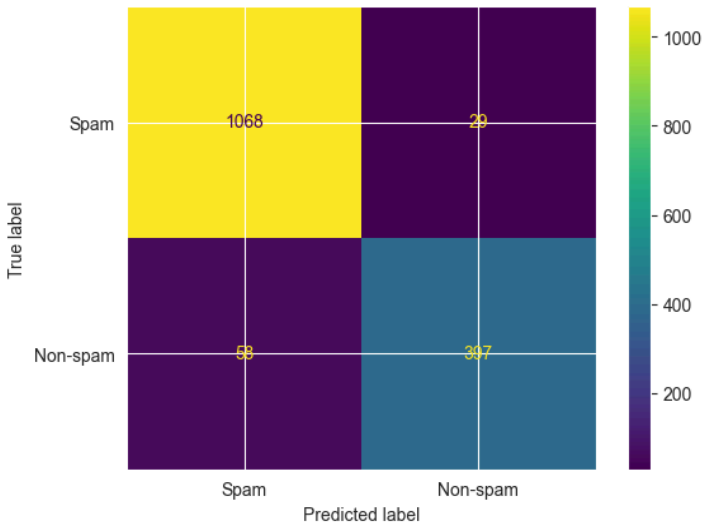
  This time we tried to apply the same function we applied previously on the previous models for the hyperparameter “var_smoothing”, but the output was 1.23284e-09, which is so small. So, we ignored this hyperparameter, and built a standard GNB model.

Applications Project

Here are the results we got:

```
Precision :  0.931924882629108

Recall :  0.8725274725274725

Accuracy Score :  0.9439432989690721

F1 Score :  0.9012485811577753

 [[1068   29]
 [  58  397]]
```

The Confusion Matrix Plot:



7.Model Evaluation & Results

How do we compare models? What are the various metrics used? Comparing models is an important step in machine learning to determine the performance of different models on a given task. There are various metrics used to evaluate the performance of a model, depending on the type of task and the nature of the data. Some commonly used metrics are:

Confusion Matrix: A confusion matrix is a table that is used to evaluate the performance of a classification model by comparing the predicted and actual values of the target variable. It is also known as an error matrix.

A confusion matrix consists of four components:

1. True Positives (TP): The number of instances that are correctly classified as positive.
2. False Positives (FP): The number of instances that are incorrectly classified as positive.
3. True Negatives (TN): The number of instances that are correctly classified as negative.
4. False Negatives (FN): The number of instances that are incorrectly classified as negative.

The confusion matrix is typically displayed in a table format with the predicted values along the top row and the actual values along the first column. The entries in the table represent the number of instances in each combination of predicted and actual values.

Using the values in the confusion matrix, several metrics can be calculated to evaluate the performance of a classification model, including accuracy, precision, recall, and F1-score.

Accuracy: Accuracy is a common evaluation metric used in classification tasks to measure the proportion of correctly classified instances among all instances in the dataset. It is calculated as the ratio of the number of correct predictions to the total number of predictions made by the model.

Mathematically, accuracy can be expressed as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ prediction} \times 100\%$$

In simpler terms, accuracy answers the question: "What proportion of the predictions made by the model are correct?"

Applications Project

Precision: Precision is a metric used in binary classification tasks to measure the proportion of correctly predicted positive instances (true positives) among all instances predicted as positive by the model. It focuses on the accuracy of the positive predictions made by the model.

Precision is calculated as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

In simpler terms, precision answers the question: "Of all the instances predicted as positive by the model, how many were actually positive?"

Recall: Recall, also known as sensitivity or true positive rate, is a metric used in binary classification tasks to measure the proportion of correctly predicted positive instances (true positives) out of all actual positive instances in the dataset. It focuses on the model's ability to correctly identify all positive instances, regardless of any false positives.

Recall is calculated as:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

In simpler terms, recall answers the question: "Of all the actual positive instances in the dataset, how many did the model correctly identify?"

F1-score: The F1-score is a metric used to evaluate the performance of a binary classification model, taking into account both precision and recall. It is the harmonic mean of precision and recall, providing a single score that balances the trade-off between these two metrics.

The formula for calculating the F1-score is:

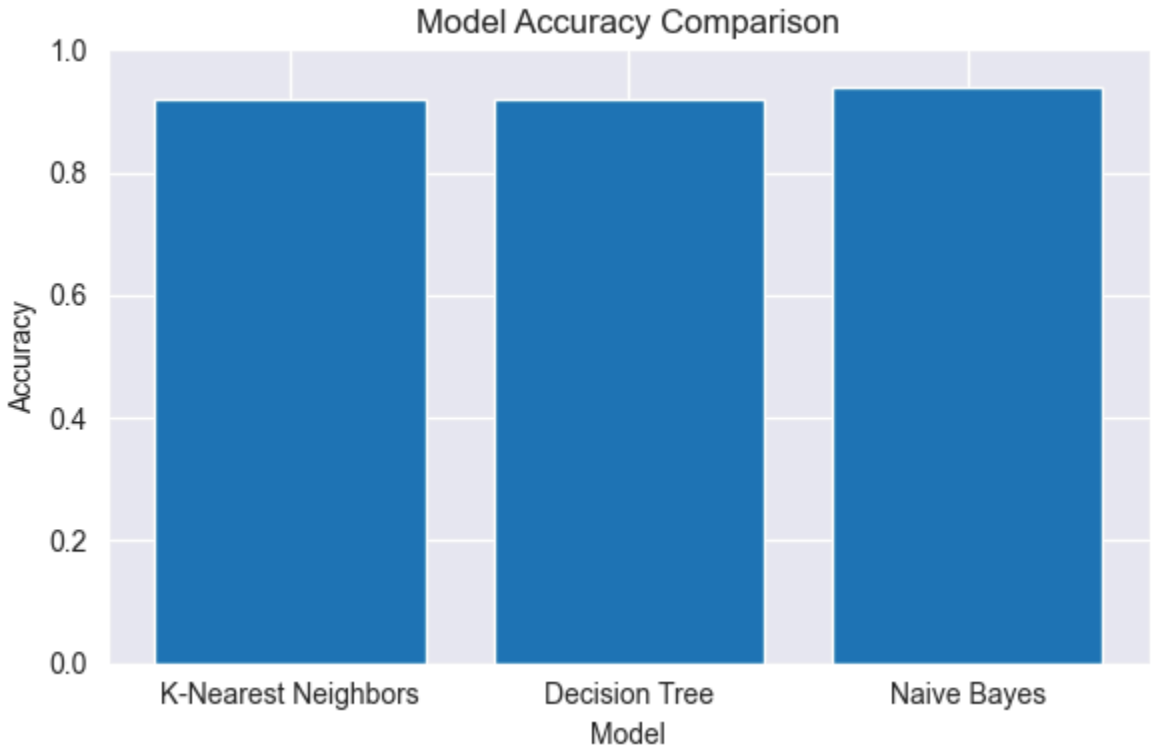
$$F1\_Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In simpler terms, the F1-score represents the balance between precision and recall, with values ranging from 0 to 1. A high F1-score indicates that the model has both high precision and high recall, meaning it has a good balance between correctly identifying positive instances and avoiding false positives.

8.Inferences & conclusion

Objective of this project is to find a model that classifies an email as spam or not spam with better accuracy. The given dataset is trained using three models KNN, DecisionTree, and Naïve Bayes.

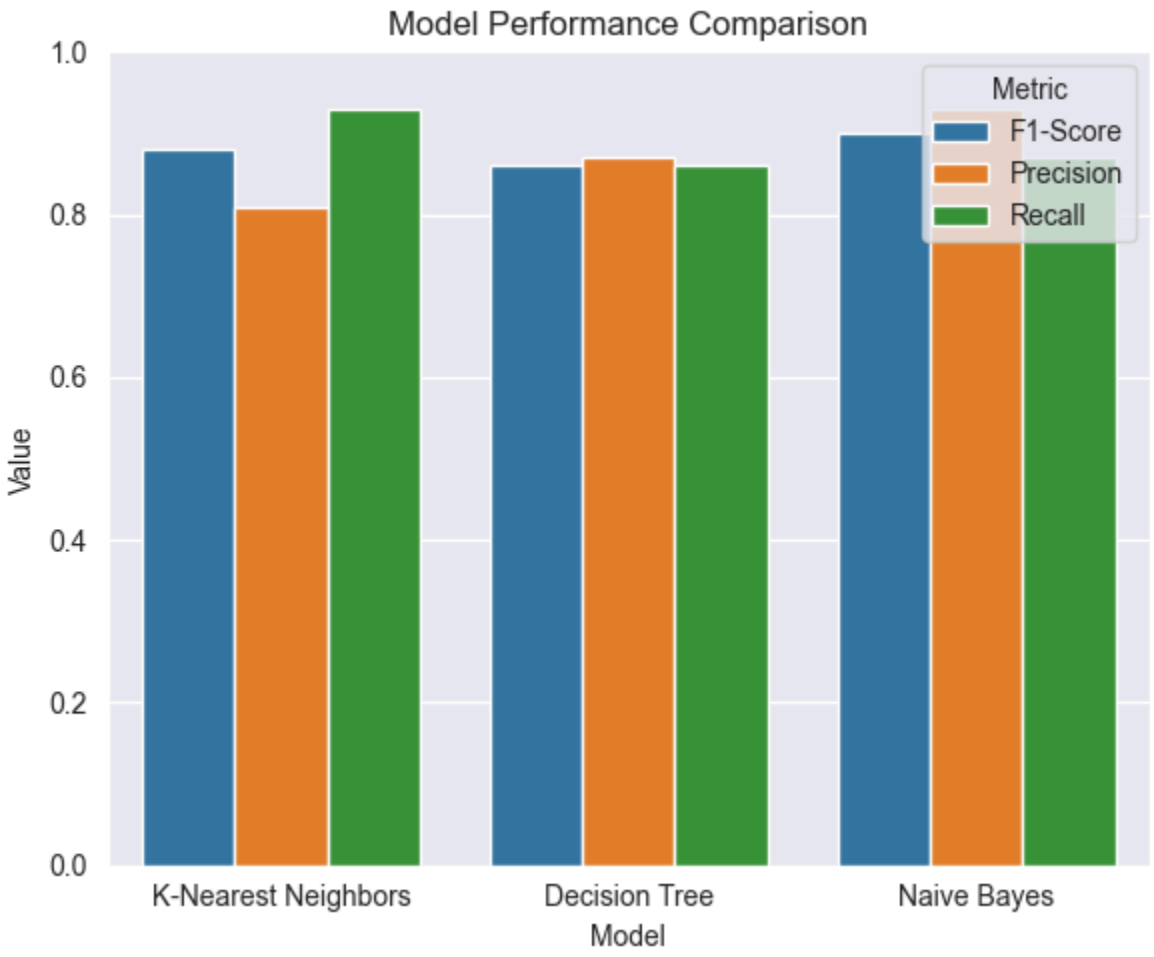
Models Accuracy Comparison:



الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Applications Project

Models Performance Comparison:



Applications Project

All the three models resulted in high accuracy , which is quite good. The following table is a comparison table that compares all the three models to find a model better than all.

Model	KNN	Decision Tree	Naive Bayse
Precision	0.81	0.87	0.93
Recall	0.93	0.86	0.87
Accuracy	0.92	0.92	0.94
F1 Score	0.87	0.87	0.90

After comparing all the attributes, it is clear that Naïve Bayes is the best model among all the models with better accuracy, precision, recall and F1 score.

Applying New Data to The Best Model

After we built and evaluated the three models, we found out that the Gaussian Naive Bayse is the best one. So, now we have to test it on a new dataframe.

First of all, we created a list called “new_emails” and filled it with 100 spam and non-spam emails. Then, we did the following preprocessing functions:

- **Tokenizing & Counting Words’ Frequencies in Each Email:**

  We used the “Counter” function from the “Collection” library, and imported the “re” library. Then, we defined the following function that tokenizes and counts words’ frequencies in each email:

```
def count_word_frequency(emails):
    word_frequencies = {}
    word_pattern = re.compile(r'\b\w+\b')
    for i, email in enumerate(emails, 1):
        email_tokens = word_pattern.findall(email.lower())
        word_counts = Counter(email_tokens)
        word_frequencies[f"Email {i}"] = word_counts
    return word_frequencies

word_frequencies = count_word_frequency(new_emails)

for email_num, frequencies in word_frequencies.items():
    print(f" {email_num}: {frequencies}")
```

- **Converting the List Into a Dataframe:**

  As we all know, machine learning models only accept dataframes as an input. So, we had to convert the list into a dataframe called “new_emails_df” using the following function:

```
new_emails_df = pd.DataFrame.from_dict(word_frequencies, orient='index').fillna(0)
new_emails_df = new_emails_df.reindex(sorted(new_emails_df.columns), axis=1)
new_emails_df.reset_index(drop=True, inplace=True)
new_emails_df.head()
```

- **Adding a Label Column to the New Dataframe:**

  In addition to creating the dataframe, we had to add a label column to train the model on it, because in our case we are using

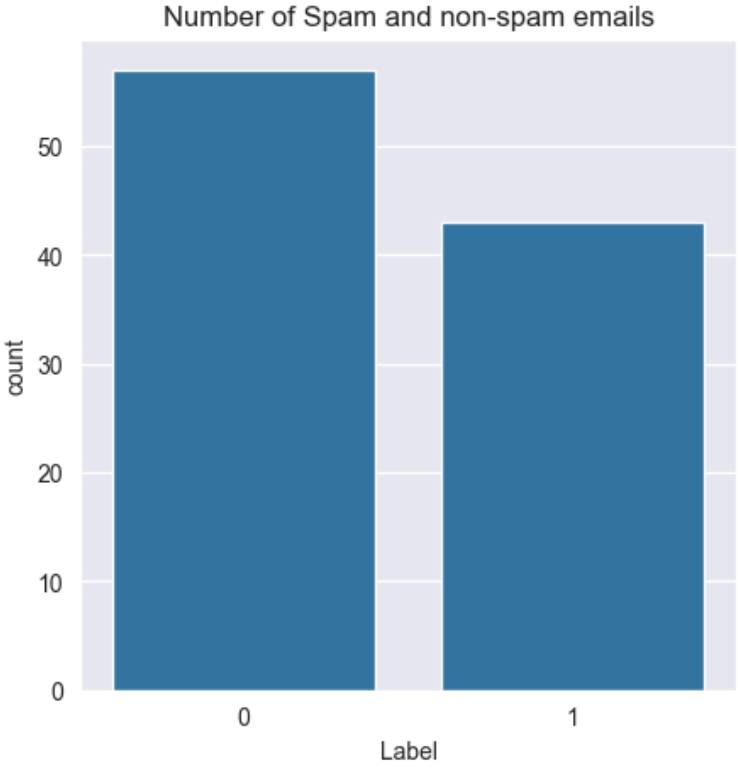
supervised learning. So, we added a column that has the value of “1” for the spam emails, and “0” for the non spam emails, and called the column “Label”.

- **Analyzing The New Dataframe:**

  - Number of rows and columns: (100 , 1080).
  - Missing or Null values: 0.
  - Duplicate rows: 0.
  - Duplicate columns: 0.
  - Datatype of records: int64.

  Here is the plot of number of spam and non-spam emails in the new dataframe:

Applications Project



- **Removing Unwanted Columns:**

  Cleaning the dataframe from unwanted and noisy columns is an essential preprocessing step to have the best performance. So, we removed the stopwords. Also, we had some digit columns' names so we had to remove them as well. Here are the functions we used:

Applying the Stopwords Function To the New Dataframe

```
new_emails_df = remove_stopwords(new_emails_df)
new_emails_df.head()
```

Removing the Digit Columns from the Dataframe

```
columns_to_keep = [col for col in new_emails_df.columns if not col.isdigit()]
new_emails_df = new_emails_df[columns_to_keep]
new_emails_df.head()
```

After applying the previously mentioned functions, we removed all the unwanted and noisy columns, and reduced the number of columns from 1080 to 971.

- **Feature Selection:**

As we mentioned previously in this documentation, feature selection is one of the most important steps of the preprocessing. So, we applied the same function we defined previously but we changed the parameters obviously. The parameters are: X, the features of the

dataframe, Y, the label of each email, “f_classif” criteria and chooses 665 features instead of 971.

- **Train-Test Split:**

  We splitted the data into “X_train”, “X_test”, “Y_train” and “Y_test” with 30% test size, and random state with the value of “42”.

- **Dataframe Scaling:**

  We scaled the dataframe to get the best performance out of our model using the following statements:

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Now, after we applied multiple data preprocessing functions, we can build and train our Gaussian Naive Bayse model and check its performance.

We built the model with no hyperparameters included, and used the “fit” statement to train it on the train subset of our dataframe.

Applications Project

```
gnb = GaussianNB()
gnb.fit(X_train, Y_train)
```

After building the model, we want to check its performance. So, we defined a function that calculates all the metrics needed to evaluate the model, such as accuracy, precision, recall and confusion matrix, and it plots the confusion matrix as well.

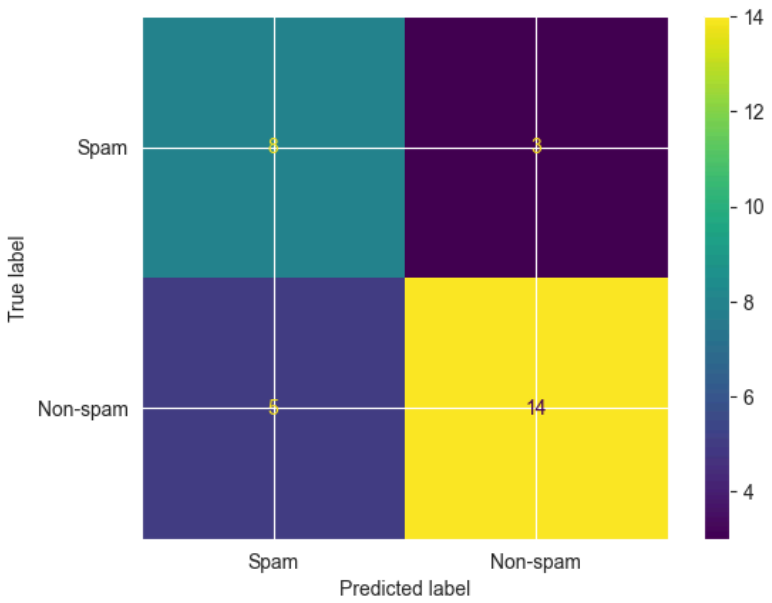
```
def perform_Test(y_pred):
    print("\nPrecision : ", precision_score(Y_test, y_pred))
    print("\nRecall : ", recall_score(Y_test, y_pred))
    print("\nAccuracy Score : ", accuracy_score(Y_test, y_pred))
    print("\nF1 Score : ", f1_score(Y_test, y_pred))
    print("\n", confusion_matrix(Y_test, y_pred))
    print("")
    cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_matrix(Y_test, y_pred), display_labels=['Spam', 'Non-spam'] )
    cm_display.plot()
    plt.show()
```

Here are the results we got:

Applications Project

```
Precision :  0.8235294117647058

Recall :  0.7368421052631579

Accuracy Score :  0.7333333333333333

F1 Score :  0.7777777777777778

 [[ 8  3]
 [ 5 14]]
```

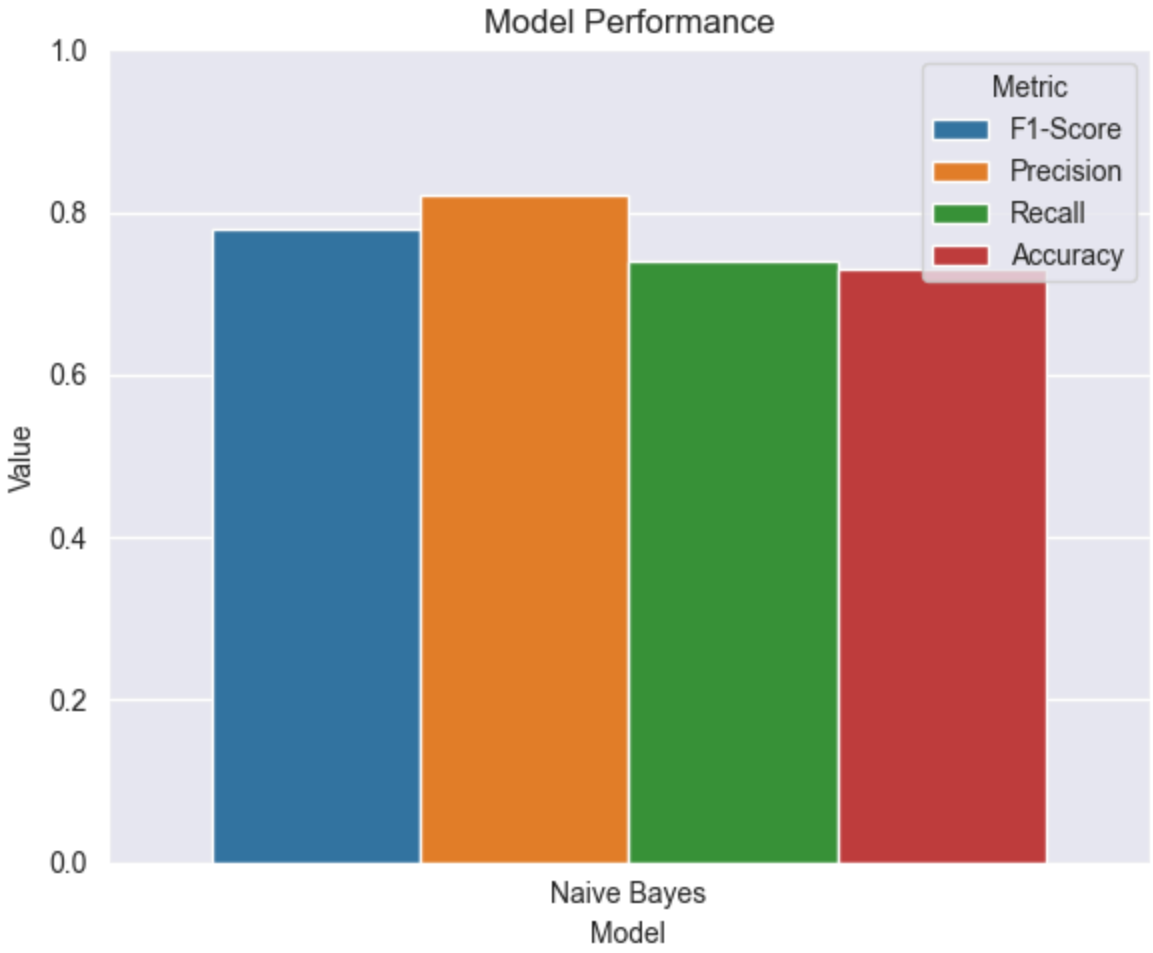
Here is the confusion matrix plot:



Finally, here is the metrics barplot:

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Applications Project



Supervisor: Dr. Maisaa Abu Qassem **Submitted By: Owais Hilal 4200051 41**

10.References

- Hindawi, Computational Intelligence and Neuroscience, “Efficient E-Mail Spam Detection Strategy Using Genetic Decision Tree Processing”.
- International Journal of Computer Applications, “Designing Spam Model- Classification Analysis”.
- ITM Web of Conferences, “Detecting Spam Emails: A Comparison Study”.
- EasyChair Preprint, “Email Spam Classification Using Machine Learning .”
- Baltic Journal Of Law & Politics, “Higher Accuracy of Spam Mail Prediction using Decision Tree Algorithm Comparing with K-Nearest Neighbor Algorithm.”
- Asian Journal of Advances in Research, “Machine Learning for SPAM Detection.”
- Acharya Narendra Dev. College, University of Delhi, “Email Spam Detection.”