Building a Smarter AI-Powered Spam Classifier

Phase-4 Document Submission

Project Overview

Project Name: Building a Smarter AI-Powered Spam Classifier Project

Project Phase: Phase 4 – Development Part 2

Phase Overview

In Phase 4 of the project, Building a smarter AI-powered spam classifier, feature engineering is the process of selecting and creating relevant input variables (features) that enable the model to distinguish between spam and legitimate messages effectively. Model training involves using a machine learning algorithm to teach the system to recognize patterns in the data and make predictions. Evaluation is the critical step where the model's performance is assessed using metrics like precision, recall, and F1-score to ensure it can accurately classify spam while minimizing false positives. These three steps collectively improve the classifier's accuracy and efficiency in identifying and filtering out spam messages.

Model training

Model training is a pivotal phase in the development of AI-powered spam classifiers. During this stage,

- The selected machine learning or deep learning algorithm is exposed to the labeled dataset of spam and non-spam messages.
- The model learns to recognize patterns and features within the data, enabling it to make predictions on new, unseen messages.

- Training involves the adjustment of model parameters, often using optimization techniques, to minimize the difference between its predictions and the actual labels in the training data.
- Fine-tuning hyperparameters and implementing regularization techniques help ensure the model generalizes well to new, unseen data.
- The success of a spam classifier heavily depends on the quality and size of the training dataset, the choice of algorithm, and the iterative process of training and validation to achieve optimal performance.

1. Prepare the data: This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.

2. Split the data into training and test sets: The training set will be used to train the model, and the test set will be used to evaluate the perform of the model on unseen data.

3. Choose a machine learning algorithm: There are a number of machine learning algorithms that can be used for Spam Classifier. MultinomialNB Regression, Random Regression.

4. Tune the hyperparameters of the algorithm: The hyperparameters of a machine learning algorithm are parameters that control the learning process. It is important to tune the hyperparameters of the algorithm to optimize its performance.

5. Train the model on the training set: This involves feeding the training data to the model and allowing it to learn the relationships between the features and spam classifier.

6. Evaluate the model on the test set: This involves feeding the test data to the model and measuring how well it predicts the spam and hum.

If the model performs well on the test set, then you can be confident that it will generalize well to new data.

Data Preparation

- Gather a large and diverse dataset of emails or messages that are labeled as either spam or not spam (ham). Make sure your dataset represents real-world scenarios.

- We performed further text preprocessing and cleaning using NLP techniques to prepare the text data for analysis. We performed further text preprocessing and cleaning using NLP techniques to prepare the text data for analysis.

- We started by loading the pre-processed data, which had been cleaned and prepared in the previous phase.

Program:

```
# Import Lib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
# importing Stopwords
import nltk
from nltk.corpus import stopwords
import string
# models
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
# train test split
from sklearn.model_selection import train_test_split
# Pipeline
from sklearn.pipeline import Pipeline
# score
from sklearn.metrics import
confusion_matrix,classification_report,ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
# Import Data
df=pd.read_csv('preprocessed_data.csv')
```

Data Splitting and Training and Test

- Split your dataset into training, validation, and test sets. The training set is used to train the model, the validation set helps tune hyperparameters, and the test set is for final model evaluation.

Program:

```
sns.histplot(data)
plt.show()


# define X(features),y(target)
X= df['text']
y=df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
```

Model Building

- **Choose a machine learning algorithm:** There are a number of machine learning algorithms that can be used for Spam Classifier. MultinomialNB Regression, Random Regression.

- Experiment with various hyperparameters to optimize the model's performance. You can use techniques like cross-validation to fine-tune these hyperparameters.

MultinomialNB Regression

Program:

```
lin_model =  # Your linear model
mnb_model = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process)),
    ('tf', TfidfTransformer()),
    ('classifier', MultinomialNB())

# fit the data
lin_model.fit(X_train, y_train)
mnb_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

# predict the target feature
lin_predict = lin_model.predict(X_test)
mnb_predict = mnb_model.predict(X_test)
rf_predict = rf_model.predict(X_test)
print(classification_report(y_test,pred_mnb))
```

Random Forest Regression

Program:

```
# Your Random Forest regression model
rf_model = RandomForestRegressor()

# fit the data
rf_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

# predict the target feature
rf_predict = lin_model.predict(X_test)
mnb_predict = mnb_model.predict(X_test)
rf_predict = rf_model.predict(X_test)
print(classification_report(y_test,pred_mnb))
```

Evaluation

- Model performance is assessed using two primary metrics: Root Mean Squared Error (RMSE) and R-squared (R2). RMSE measures the accuracy of spam and ham predictions, while R2 quantifies how well the model fits the data.

- Cross-validation is also performed to evaluate model accuracy using the cross_val_scorefunction.

- ❖ Assess the model's performance on the validation set using appropriate evaluation metrics, such as accuracy, precision, recall, F1-score, and ROC AUC.

- ❖ Make necessary adjustments to the model based on the validation results.

Program:

```
performance = [
   ["LinearModel", performance(y_test, lin_predict)[0], performance(y_test, lin_predict)[1], crossval(lin_model)],
   ["MultinomialNB", performance(y_test, mnb_predict)[0], performance(y_test, mnb_predict)[1], crossval(mnb_model)],
   ["RandomForest", performance(y_test, rf_predict)[0], performance(y_test, rf_predict)[1], crossval(rf_model)]
]

pd.options.display.float_format= '{:,.2f}'.format
conclusions=pd.DataFrame(performance,columns=["Model","RMSE","R2","Accuracy"])

print(conclusions)
```

Conclusion

In this phase of the project, building a smarter AI-powered spam classifier is a complex and iterative process that requires careful consideration of various factors. The success of such a project hinge on data quality, model selection, training, and deployment strategies, as well as ethical and legal considerations. It's important to highlight that the fight against spam is an ongoing battle, and the project should be designed to adapt to evolving tactics used by spammers. By following best practices and staying committed to improving and maintaining the system, you can create a highly effective spam classifier that enhances user experience, privacy, and security while minimizing false positives. The project represents a multifaceted endeavor that combines data science, machine learning, engineering, and user engagement to address a persistent and ever-evolving challenge in the digital world.