T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4288 Introduction to Machine Learning

Term Project Model Development Report

Group Members

Student ID	Name Surname	Contact
150120013	İrem Aydın	irmaydin14@gmail.com
150120049	Aksanur Konuk	aksanurkonuk@gmail.com
150120054	Elife Kocabey	kocabeyelife@gmail.com
150120066	Zeynep Yılmaz	zenepyilmaz66@gmail.com
150121013	İrem Kıranmezar	iremkiranmezar@gmail.com

Introduction

In the Model Development phase of our Sentiment Analysis project, we trained our dataset with 8 models and obtained the results of this training. These models, which will be detailed in the Models for Training section, are Supervised Learning and Deep Learning models. Since the computers we had during training were insufficient, we used 10 thousand rows of the 1.6 million data in the data set in model training.

We split the dataset into training and test sets, ensuring that 20% of the data was used for testing while 80% was reserved for training. This setup allows us to assess the model's performance on unseen data, ensuring its generalizability.

Code Explanation and Model Training

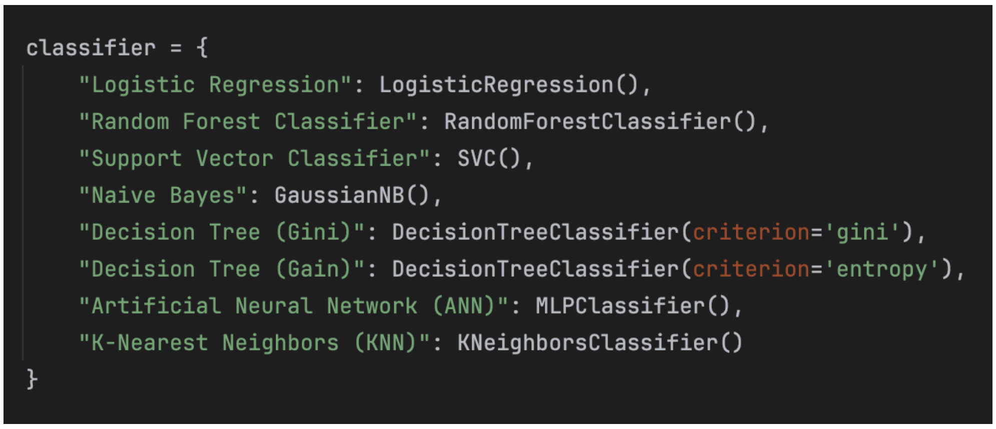
The code begins by preparing the data, where the target variable y is separated from the feature set X. This is followed by splitting the data into training and test sets using the train_test_split function. A test size of 20% is chosen, with the remaining 80% used for training. This split ensures that the models are trained on a substantial portion of the data and then tested on unseen data to evaluate their performance. The shapes of the training and test sets are printed for confirmation.

The next step involves converting the data into an appropriate format. If the training or test data is in a sparse matrix format (common when working with text data preprocessed using methods like TF-IDF or CountVectorizer), it is converted into an array format using the toarray() method. This is important because many machine learning models require input data to be in a dense array format to function properly.

The classifiers used in the project are defined in a dictionary. This dictionary contains various machine learning models, including **Logistic Regression**, **Random Forest Classifier**, **Support Vector Classifier (SVC)**, **Naive Bayes**, **Decision Tree (Gini)**, **Decision Tree (Gain)**, **K-Nearest Neighbour** and **Artificial Neural Network (ANN)**. Each model is instantiated and stored with its corresponding name, making it easier to loop through them for training and evaluation.

For each model, the following steps are performed: the model is trained using the training data (x_train, y_train) with the fit() method, and predictions are made on the test data (x_test) using the predict() method. The performance of each model is evaluated using several metrics. The **accuracy** of the model is calculated using the accuracy_score function. Additionally, the **confusion matrix** is generated using confusion_matrix, which provides insight into how well the model classified the data. The **classification report** is produced using classification_report, providing detailed performance metrics such as precision, recall, and F1-score.

Furthermore, cross-validation scores are calculated using the cross_validate function with 10-fold cross-validation. This provides an indication of how well the model performs on different subsets of the training data. The mean scores for accuracy, precision, recall, and F1 are displayed to give an overall view of the model's robustness.



```
classifier = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest Classifier": RandomForestClassifier(),
    "Support Vector Classifier": SVC(),
    "Naive Bayes": GaussianNB(),
    "Decision Tree (Gini)": DecisionTreeClassifier(criterion='gini'),
    "Decision Tree (Gain)": DecisionTreeClassifier(criterion='entropy'),
    "Artificial Neural Network (ANN)": MLPClassifier(),
    "K-Nearest Neighbors (KNN)": KNeighborsClassifier()
}
```

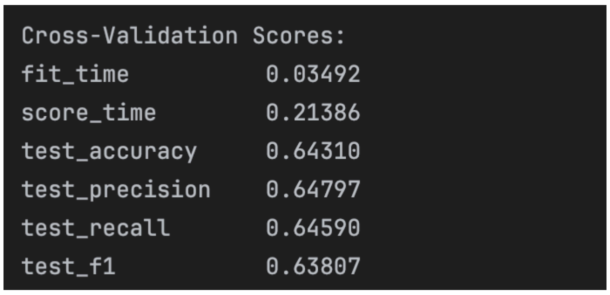
Model Outputs

After training and evaluating each model, several key results are displayed for each classifier. First, the **accuracy score** is provided, which represents the proportion of correct predictions made by the model. The **confusion matrix** is shown next, which indicates the number of true positive, true negative, false positive, and false negative predictions, offering a deeper understanding of the model's classification behavior. The **classification report** follows, summarizing key performance metrics, including precision, recall, and F1-score for both classes (positive and negative sentiment). Finally, the **cross-validation scores** are presented, which provide the mean accuracy, precision, recall, and F1-scores from 10-fold cross-validation. This helps assess how well the model generalizes to unseen data.

Challenges and Solutions

One of the main challenges encountered during the project was the size of the dataset, which contained 1.6 million records. Such a large dataset could create problems in terms of memory usage and processing time. Therefore, instead of the entire dataset, a subset of 10,000 records was used, selected to keep the ratio of positive and negative labels in balance. This solution accelerated the data processing and model training process, providing a more efficient work and continuing to represent the general features of the large dataset. In this way, the difficulties experienced due to the size of the dataset were overcome.

We used techniques like cross-validation to optimize the models and improve their accuracy.



```
Cross-Validation Scores:
fit_time          0.03492
score_time        0.21386
test_accuracy     0.64310
test_precision    0.64797
test_recall       0.64590
test_f1           0.63807
```

Conclusion

In the next stage, we will compare the results we obtained from the model training we added in the third stage of our project, select the most appropriate model for our project and train our data with that model.