Information Retrieval Course - Mini Project 6 Report

Delaram Fartoot - 610399153
Ashkan Zarkhah - 610399196
Narges Ghanei - 610399155
Kimia Esmaili - 610398193

(More detail about the model is commented on in the code.)

• Introduction:

Author identification in Persian literature involves the task of attributing a text to a specific author based on their writing style, linguistic features, and unique characteristics. This problem is challenging due to the morphological complexity of the Persian language, limited linguistic resources, lack of standardized datasets, and the need to consider cultural and historical context. By developing accurate author identification models, researchers can enhance our understanding of Persian literature, analyze authorship patterns, and contribute to the field of natural language processing in the Persian language.

There are several challenges to *Author Identification in Persian Literature* problem due to the complexities of the Persian language and the lack of comprehensive datasets and resources. addressing these challenges in author identification in Persian literature and Persian information retrieval and NLP requires collaboration between researchers, linguists, and computer scientists to develop robust datasets, linguistic resources, and machine learning models that can accurately identify authors based on their writing style and linguistic features. Some of the key challenges include:

Lack of standardized datasets: One of the major challenges in author identification in Persian literature is the limited availability of standardized datasets for training and testing machine learning models. This makes it difficult to accurately classify and identify authors based on their writing style and linguistic features.

Morphological complexity: The Persian language is highly inflectional and morphologically complex, which poses challenges in feature extraction and analysis for author identification. Different authors may use variations in word forms, verb conjugations, and sentence structures, making it difficult to accurately attribute a text to a specific author.

Limited linguistic resources: Another challenge in author identification in Persian literature is the scarcity of linguistic resources and tools for processing Persian text. There is a lack of comprehensive dictionaries, part-of-speech taggers, and named entity recognition tools for the Persian language, which hinders the development of accurate author identification models.

Cultural and historical context: Author identification in Persian literature also requires an understanding of the cultural and historical context in which the texts were written. Authors may use specific references, idioms, and stylistic choices that are unique to their time period or literary tradition, making it challenging to accurately attribute a text to a specific author without considering these factors.

Transfer learning and fine-tuning: In the field of NLP, transfer learning techniques have been widely used to improve the performance of author identification models. However, transferring models trained on English or other languages to Persian text may not yield optimal results due to the linguistic differences between languages. Fine-tuning pre-trained models on Persian text is crucial for achieving accurate author identification results.

In this mini-project, we are trying to make a robust, efficient, and accurate dataset and model for this task.

• Dataset Construction Techniques:

Our dataset consists of the works of Hafez, Sa’adi, Molavi, Naser Khosrow, Attar, Parvin Etesami, Shahryar, Khaghani, IrajMirza, and Ferdowsi. There are about 30 documents for each author and poet, each with around 500 words. The data is stored in a .csv file for ease of work.

We tried to use authors who wrote in different genres, content, poetic forms, and literary formats. Most of the authors of our choice used different literary formats and if there are authors with the same poetic form, they are mostly easy to distinguish (e.g., Ferdowsi and Attar both write “Masnavi”, however Ferdowsi’s works are literary epics and Attar’s works are related to spirituality.).

Most of our dataset is scraped from Ganjoor, which is an online collection of Persian poets' poems. On this website, users can search based on a poet or poem. The website is non-profit, publishes content freely, and is managed with the help of public contributions. The sources used are also freely available to everyone.

History Initially, this website existed as a subset of the Designing website. Its original name was "Ganj Adab" and included the complete works of Hafez, Khayyam's quatrains, Ferdowsi's Shahnameh, and Rumi's Mathnawi. The poems were entered into the website using the software "Duraj." Due to copyright issues with the software, the decision was made to switch to the free and open-source collection "Rira" for the initial core of Ganjoor. After transferring the "Rira" collection to the Ganjoor domain, the website began to expand.

Operation The "Rira" collection consisted of a large number of Persian poems, making it easy to form Ganjoor. To expand Ganjoor, the OCR method was used. In this system, each line of poetry is scanned using a scanner and the image is uploaded to the website. Persian speakers then enter the text they see on the website. At least three stages are completed for each line of poetry to minimize errors in the process.

Here is a description of the works of each author and specific characteristics that are helpful to identify them:

- Hafez’s works are celebrated for their profound insights, lyrical beauty, and spiritual themes.
  Genre: Persian classical poetry, particularly in the form of ghazals. Hafez's ghazals often convey deep emotions and philosophical reflections.
  Characteristics: richness of imagery, emotional depth, and intricate wordplay. His verses are known for their intricate metaphors, allusions to Sufi mysticism, and profound philosophical insights. They often have themes of love, longing, wine (metaphorically for spiritual intoxication), and the search for divine truth. They are also marked by a sense of ambiguity and multiple layers of meaning.
  Style: distinguished by its musicality, elegance, and lyrical beauty. Utilizing intricate rhyme schemes, meter, and language to create a harmonious flow of words which are characterized by their emotional intensity, sensual imagery, and spiritual resonance.

- Sa'adi was known for his philosophical reflections on human nature, ethics, and morality.
  Genre: his poetry encompasses a wide range of genres, including ghazals, qasidas, and rubaiya. They often convey profound moral lessons and ethical teachings.
  Characteristics: moral and ethical teachings, which are often presented through poignant anecdotes, allegorical tales, and wise aphorisms that are imbued with a sense of empathy, compassion, and a deep understanding of

human nature which reflect a strong sense of social justice, universal brotherhood, and the importance of kindness, generosity, and humility.
Style: They are marked by their simplicity, elegance, and clarity of expression. Straightforward language, vivid imagery, and rhythmic flow make his poems accessible. Sa'adi's verses often convey deep emotional resonance, philosophical depth, and a timeless quality.

- Molavi's works have had a profound influence on mystical and spiritual thought.
  Genre: his poetry falls within the genre of Sufi poetry, particularly in the form of ghazals, qasidas, and rubaiya, and are characterized by their exploration of spiritual themes, love, mysticism, and the quest for divine union which often convey deep spiritual insights, profound philosophical reflections, and a sense of divine ecstasy and transcendence.
  Characteristics: mystical and spiritual depth, profound insights into the nature of existence, and the quest for spiritual enlightenment. His works are known for their themes of love, unity, oneness with God, and the journey of the soul towards spiritual awakening. They often convey a sense of divine love, ecstatic rapture, and the transformative power of spiritual illumination.
  Style: is marked by its emotive language, rich symbolism, and profound metaphors. He combines elements of Persian poetry, Sufi mysticism, and Islamic philosophy which are characterized by their musicality, rhythm, and lyrical beauty.

- Naser Khosrow is known for his contributions to Persian literature in the fields of poetry, philosophy, and religious thought.
  Genre: His poetry falls within the genre of classical Persian poetry, with an emphasis on philosophical and mystical themes which include a variety of

poetic forms, such as qasidas, ghazals, and rubaiyat, which explore themes of love, spirituality, ethics, and the human experience.
Characteristics: philosophical depth, intellectual rigor, and spiritual insights which reflect a deep engagement with metaphysical questions, ethical dilemmas, and the quest for spiritual enlightenment. They convey a sense of introspection, contemplation, and a search for higher truths.
Style: is marked by its clarity, precision, and intellectual sophistication. He is known for his use of classical Persian language, intricate wordplay, and profound symbolism, which lend his verses a timeless quality and universal appeal. They are characterized by their philosophical reflections, moral teachings, and a sense of intellectual inquiry that transcends cultural and temporal boundaries.

- Attar is considered as a prominent figure in Sufi poetry.
  Genre: His verses fall within the genre of Sufi poetry, particularly in the form of ghazals, qasidas, and mystical allegorical narratives. Sufi poetry is characterized by its exploration of spiritual themes, divine love, mysticism, and the quest for spiritual enlightenment. His works often convey deep spiritual insights, moral teachings, and the transformative power of mystical experiences.
  Characteristics: is characterized by its profound spiritual depth, allegorical storytelling, and moral teachings, which often reflect Sufi concepts such as divine love, the journey of the soul towards union with the Divine, and the importance of self-realization and spiritual enlightenment. They are known for their rich symbolism, vivid imagery, and moral allegories that convey profound spiritual truths.
  Style: lyrical beauty, emotional intensity, and allegorical storytelling. He weaves together elements of Persian poetry, Sufi mysticism, and spiritual symbolism. They are characterized by their rich imagery, metaphorical language, and intricate wordplay.

Challenges Encountered in Dataset Creation and Addressing Them:

- Limited availability of Persian literary works in digital format: Collaborating with libraries and cultural institutions to digitize and make more works available online.
- Ensuring dataset diversity: Actively seeking out works from a variety of authors and genres to create a comprehensive dataset.
- Data annotation and cleaning: Implementing automated tools for data cleaning and annotation, as well as involving experts in literature for manual verification.
- Data privacy and copyright concerns: Obtaining proper permissions and licenses for using copyrighted works, and ensuring data privacy regulations are followed.
- Balancing dataset size and quality: Continuously updating and refining the dataset to include more work while maintaining high-quality standards.

• Model Selection and Fine-Tuning

Choice of model:

As mentioned earlier, one of the concerns we have is that pre-trained models are primarily designed for English and English-based languages, making them less adapted to Farsi, especially considering the intricacies of poems written in Persian. Most pre-trained models are trained on day-to-day English language data, which poses a challenge for handling the complexities of the Persian language, especially in the context of poetry. To address this issue, we opted for two ParsBERT models: bert-fa-base-uncased and bert-fa-base-uncased-clf-persiannews. Several factors influenced this choice. ParsBERT models are pre-trained specifically on the Persian language, providing a significant advantage over generic BERT models, and also have performed much better in Persian text classification problems. However, it's essential to note that our expectations for performance should be tempered by the fact that our dataset is relatively small. Additionally, we are faced with the challenge of predicting 10 classes, which is considerably fewer than what is common in most classification problems.

Given the limited size of our dataset, it's plausible that classical approaches may yield better results. We will further explore these traditional methods in the upcoming sections

Fine tuning process:

We followed the same procedure for both models. The models are initialized with a BERT architecture that has been pre-trained on a large corpus. During fine-tuning, the model is exposed to task-specific labeled data, and its parameters are updated to better align with the classification problem. The BERT layers, originally trained for language understanding, capture contextual information in the input text. The additional layers, including dropout for regularization and a linear classifier tailored to the number of classes in the task, are fine-tuned to make predictions. This process enables the model to leverage its pre-existing knowledge from diverse language contexts while adjusting to the target classification task. The objective is to improve the model's performance on the specific dataset, considering factors like the limited size of the labeled data and the number of classes to predict. Fine-tuning optimizes the model for the desired classification goal, contributing to enhanced task-specific performance.

Architecture:

As mentioned earlier, we utilized two models in our approach. Specifically, we employed pretrained ParsBERT models, each augmented with two additional layers - dropout and a classifier - to construct the respective architectures. These models share several tunable parameters, such as the choice of loss function, optimizer, learning rate, scheduler, and others. The exploration of the optimal learning rate will be addressed in subsequent sections. For both models, we adopted cross-entropy as the loss function and Adam as the optimizer, while setting the batch size to 16. A detailed list of the many other parameters we selected can be found within the code.

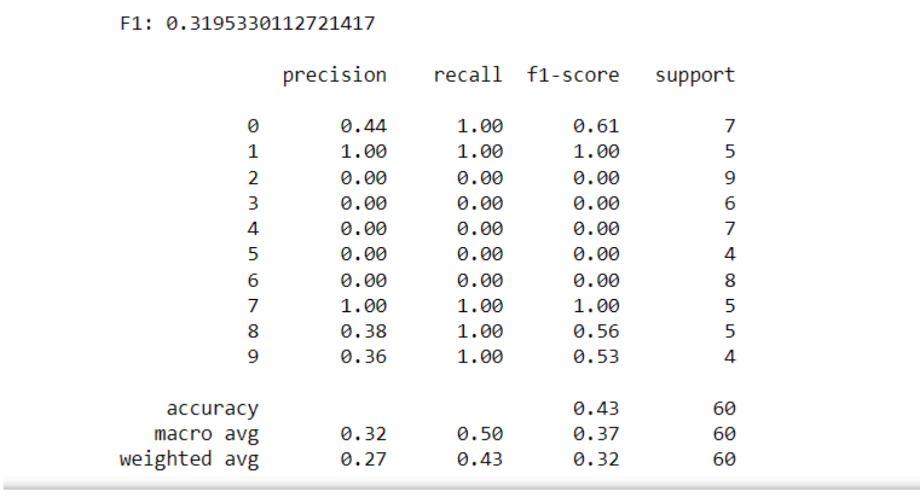
• Experiments and Results with 5-Fold Cross Validation:

5-Fold Cross-Validation:

Considering our small dataset, we aim to maximize the use of available data. To achieve this, we employ 5-fold cross-validation, wherein we divide our data into five subsets. In each iteration, one subset serves as the validation set while the remaining four act as the training set. We then compute the average metrics across these iterations to obtain our final results. This approach ensures that every data point is included in the validation set at least once, optimizing our utilization of the limited dataset. However, it's important to note that our dataset is still relatively small, and the validation set, consisting of 60 data points, may be considered small in comparison.

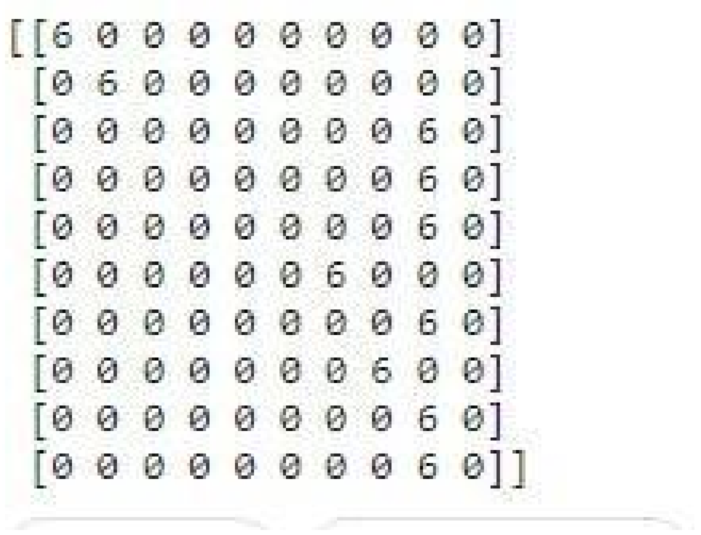
Fine-tuning:

There are various aspects we can fine-tune in our model. As mentioned earlier, we are employing 5-fold cross-validation and utilizing average accuracy and average F1 for comparison. While we showcase the confusion matrix for one iteration, our primary emphasis lies on the average F1 and average accuracy. The detailed metrics for a single iteration are as follows:



```
F1: 0.3195330112721417
              precision    recall  f1-score   support

           0       0.44      1.00      0.61         7
           1       1.00      1.00      1.00         5
           2       0.00      0.00      0.00         9
           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00         7
           5       0.00      0.00      0.00         4
           6       0.00      0.00      0.00         8
           7       1.00      1.00      1.00         5
           8       0.38      1.00      0.56         5
           9       0.36      1.00      0.53         4

    accuracy                           0.43        60
   macro avg       0.32      0.50      0.37        60
weighted avg       0.27      0.43      0.32        60
```

Additionally, the best performance for our models was achieved with stopwords removal, and the corresponding confusion matrix is as follows:



```
[[6 0 0 0 0 0 0 0 0 0]
 [0 6 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 6 0 0 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 0 6 0 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 0 0 6 0]]
```

With our best model, we have accurately predicted many classes. However, a notable flaw lies in the prediction of Class 5 (Shahryar), where the model seems to rely on the characteristics of other classes. This issue may be mitigated by acquiring more data specifically for Shahryar and other classes that have not been predicted even once. By enriching the dataset with diverse examples from these underrepresented classes, the model can improve its understanding and make more accurate predictions, addressing the current limitation

Now, to compare the metrics for different parameters, please consider that Model1 represents bert-fa-base-uncased, and Model2 represents bert-fa-base-uncased-clf-persiannews.

Learning parameter:

We experimented with two learning rates for our model performance: 2e-4 and 2e-5. The results were as follows:

	Average f1	Average Accuracy	Best f1	Best Accuracy
Model1 with alpha = 2e-5	0.247	0.348	0.43	0.57
Model1 with alpha = 2e-4	0.007	0.066	0.008	0.07
Model2 with alpha = 2e-5	0.246	0.332	0.39	0.48
Model2 with alpha = 2e-4	0.007	0.066	0.008	0.07

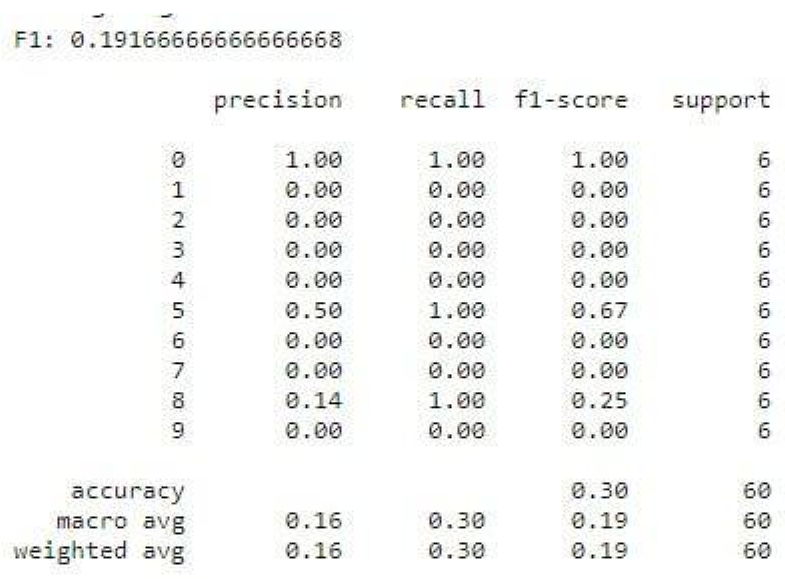
As evident from the results, when using the same learning rates for both models, their performance remains comparable. However, the noteworthy improvement observed with smaller learning rates can be attributed to the finer granularity of parameter updates during training.

Smaller learning rates allow the model to make smaller adjustments to its parameters in each iteration, facilitating a more exploration of the parameter space. This finer tuning can be particularly beneficial when navigating complex or narrow regions of the optimization landscape. On the other hand, larger learning rates might cause the model to overshoot optimal parameter values, leading to slower convergence or even divergence.

In essence, the choice of a smaller learning rate enhances the model's ability to find more precise and stable minima during the optimization process, contributing to the observed superior performance.

Document length:

As seen above the first model performs slightly better and the proper learning rate is 2e-5. Now for this part we want to experiment if shortening our data due to document length helps our performance. We chose a threshold which is the average of document lengths. The results were as follows:

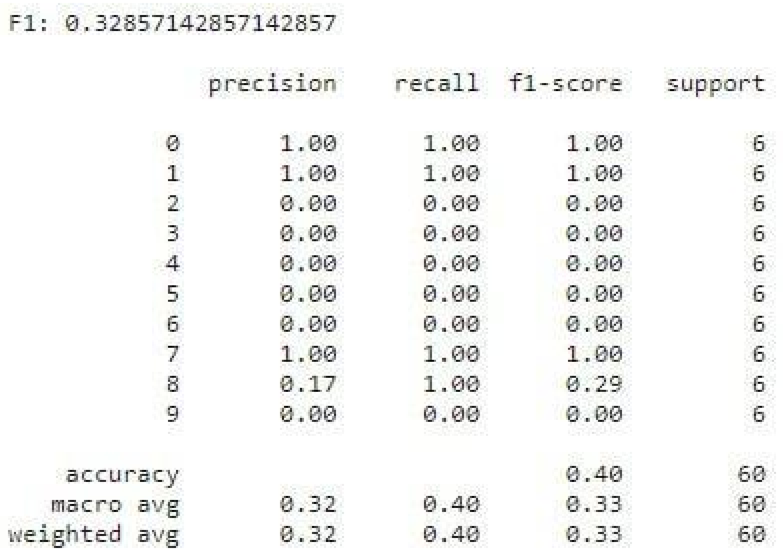


```
F1: 0.19166666666666668

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       0.00      0.00      0.00         6
           2       0.00      0.00      0.00         6
           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00         6
           5       0.50      1.00      0.67         6
           6       0.00      0.00      0.00         6
           7       0.00      0.00      0.00         6
           8       0.14      1.00      0.25         6
           9       0.00      0.00      0.00         6

    accuracy                           0.30        60
   macro avg       0.16      0.30      0.19        60
weighted avg       0.16      0.30      0.19        60
```

As you can see, removing it worsens our performance. Cutting the data results in a significant loss of information, which is undesirable, especially considering our dataset's inherent limitation in size. Typically, document shortening is employed when dealing with larger datasets to manage computational costs. However, given our current dataset's small size, we can afford the computational expenses associated with its entirety. Therefore, shortening the document is not a suitable approach in our case. If the goal is to reduce the document size, a more appropriate strategy would be to remove stopwords, as discussed in the upcoming section.

Stopwords:

In this section, we introduced some Persian stopwords to be removed from our documents. Upon comparing metrics, it is evident that this performance is the best we have achieved so far.


```
F1: 0.32857142857142857

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         6
           2       0.00      0.00      0.00         6
           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00         6
           5       0.00      0.00      0.00         6
           6       0.00      0.00      0.00         6
           7       1.00      1.00      1.00         6
           8       0.17      1.00      0.29         6
           9       0.00      0.00      0.00         6

    accuracy                           0.40        60
   macro avg       0.32      0.40      0.33        60
weighted avg       0.32      0.40      0.33        60
```

The effectiveness of removing stopwords lies in the fact that these words are commonly used and do not carry significant semantic meaning in a given context. By eliminating them, we reduce noise and focus on the more meaningful words in the text. This helps improve the overall performance of our model as it can concentrate on the content-rich terms, leading to better accuracy, precision, and other performance metrics. Essentially, removing stopwords enhances the signal-to-noise ratio in our dataset, contributing to the observed improvement in performance.

Epochs:

Since our dataset is very small, one approach is to use a higher number of epochs. However, we have observed that this tends to lower performance due to overfitting on the training data. With our limited training data, a high number of epochs can lead the model to fit too closely to that specific data, resulting in poor performance on new data. On the other hand, a small number of epochs may result in underfitting, failing to fine-tune the model with the available data and also lowering overall performance. Therefore, determining the right number of epochs remains a challenge.

• Comparison with Traditional ML Approaches

Document preparation:

In this phase, we initially retrieve the preprocessed data and proceed to tokenize each document. Following this step, we eliminate stop words from the documents using predefined Farsi stopwords available in the Hazm library. Subsequently, we create a label list that associates each document with its corresponding author label. Then, we concatenate all tokens of each document to form a single string, ensuring that this string excludes stop words.

Train and test split:

Next, we proceed to select the training and testing data. We allocate 20% of the documents to the test dataset. We designate 20% of the dataset and its corresponding labels randomly as the test data, while the remaining documents serve as the training data, each labeled accordingly.

Random Forest Model:

1. Feature Extraction:

- The TfidfVectorizer is a feature extraction method that transforms text data into numerical features using the Term Frequency-Inverse Document Frequency (TF-IDF) representation.
- max_features=1000 specifies the maximum number of features (words) to be included in the vocabulary. Only the top 1000 most frequent words will be considered.
- tfidf_vectorizer.fit_transform(X_train) computes the TF-IDF representation of the training data X_train and fits the vectorizer to the training data. It transforms the text data into a sparse matrix where rows represent documents and columns represent features (words).

- tfidf_vectorizer.transform(X_test) transforms the test data X_test using the fitted vectorizer. It ensures that the test data is transformed using the same vocabulary learned from the training data.

2. Train the Random Forest Classifier:

- The RandomForestClassifier is a machine learning algorithm based on the ensemble learning technique known as the Random Forest.
- n_estimators=100 specifies the number of decision trees in the Random Forest ensemble.
- random_state=42 ensures reproducibility of results by setting a seed for random number generation.
- rf_classifier.fit(X_train_tfidf, y_train) trains the Random Forest classifier on the TF-IDF transformed training data X_train_tfidf with corresponding labels y_train.

3. Evaluate the Model:

- rf_classifier.predict(X_test_tfidf) predicts the labels for the test data based on the trained Random Forest classifier.
- accuracy_score(y_test, rf_predictions) calculates the accuracy of the model by comparing the predicted labels (rf_predictions) with the true labels (y_test).
- classification_report(y_test, rf_predictions) generates a detailed classification report including precision, recall, F1-score, and support for each class.
- confusion_matrix(y_test, rf_predictions) computes the confusion matrix, which shows the counts of true positive, true negative, false positive, and false negative predictions for each class.

4. Print Results:

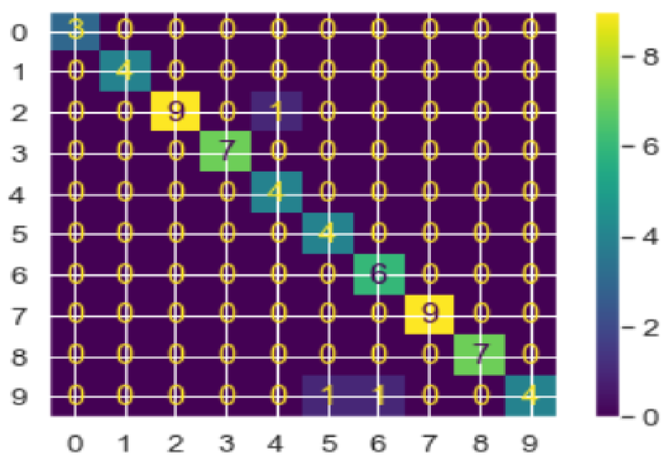
- The script prints the accuracy score, classification report, and confusion matrix to evaluate the performance of the trained model.

This model leverages TF-IDF vectorization for feature extraction and the Random Forest classifier for text classification tasks.

The results are as below:

```
Accuracy: 0.95
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         4
           2       1.00      0.90      0.95        10
           3       1.00      1.00      1.00         7
           4       0.80      1.00      0.89         4
           5       0.80      1.00      0.89         4
           6       0.86      1.00      0.92         6
           7       1.00      1.00      1.00         9
           8       1.00      1.00      1.00         7
           9       1.00      0.67      0.80         6

    accuracy                           0.95        60
   macro avg       0.95      0.96      0.94        60
weighted avg       0.96      0.95      0.95        60
```

Confusion matrix:



	0	1	2	3	4	5	6	7	8	9
0	3	0	0	0	0	0	0	0	0	0
1	0	4	0	0	0	0	0	0	0	0
2	0	0	9	0	1	0	0	0	0	0
3	0	0	0	7	0	0	0	0	0	0
4	0	0	0	0	4	0	0	0	0	0
5	0	0	0	0	0	4	0	0	0	0
6	0	0	0	0	0	0	6	0	0	0
7	0	0	0	0	0	0	0	9	0	0
8	0	0	0	0	0	0	0	0	7	0
9	0	0	0	0	0	1	1	0	0	4

SVM model:

1.Initialize and Train the SVM Classifier:

- svm_classifier = SVC(kernel='linear') initializes the SVM classifier with a linear kernel. SVMs can use different kernel functions to transform the input data into a higher-dimensional space where it can be linearly separated.
- svm_classifier.fit(X_train_tfidf, y_train) trains the SVM classifier on the TF-IDF transformed training data X_train_tfidf with corresponding labels y_train. The classifier learns the optimal hyperplane that separates the data points into different classes based on the features extracted by TF-IDF.

2. Predict Labels for Test Data:

- svm_predictions = svm_classifier.predict(X_test_tfidf) predicts the labels for the test data X_test_tfidf using the trained SVM classifier.

3. Evaluate the Model:

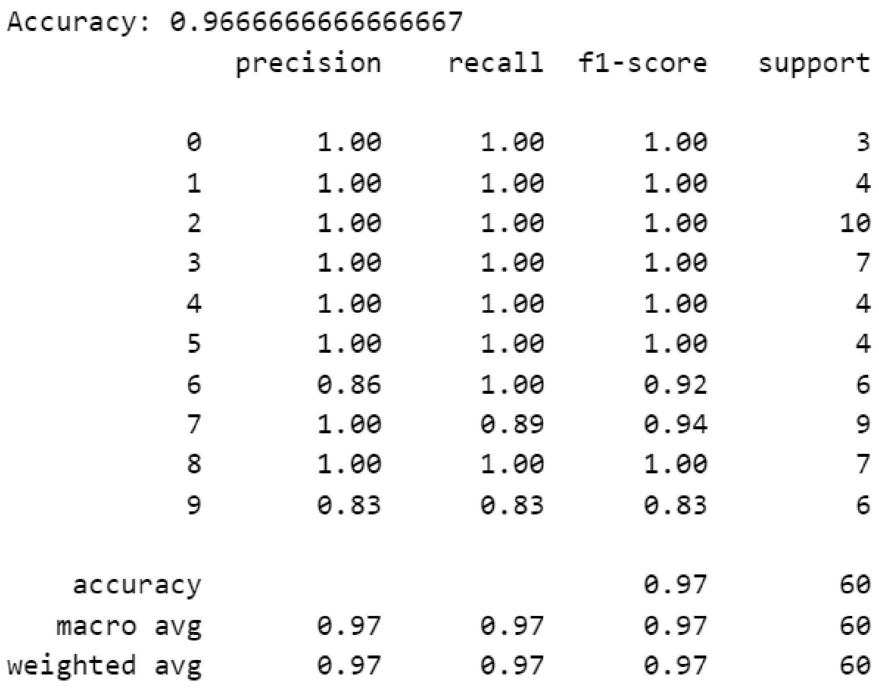
- accuracy_score(y_test, svm_predictions) calculates the accuracy of the model by comparing the predicted labels (svm_predictions) with the true labels (y_test).
- classification_report(y_test, svm_predictions) generates a detailed classification report including precision, recall, F1-score, and support for each class.
- confusion_matrix(y_test, svm_predictions) computes the confusion matrix, which shows the counts of true positive, true negative, false positive, and false negative predictions for each class.

4. Print Results:

   - The script prints the accuracy score, classification report, and confusion matrix to evaluate the performance of the trained SVM classifier.

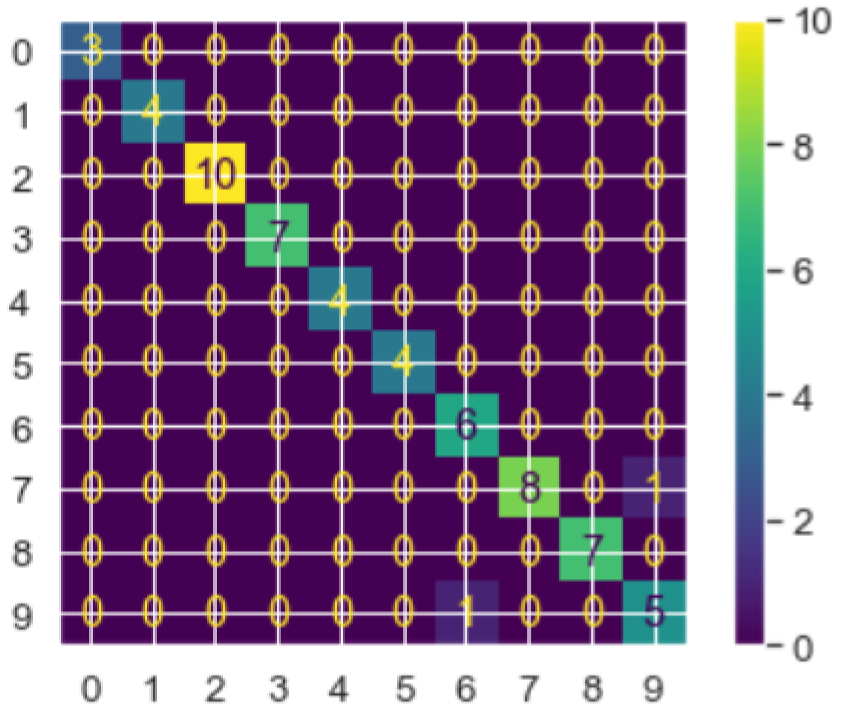
This model utilizes Support Vector Machines (SVMs) with a linear kernel to classify text data based on the TF-IDF features. SVMs are effective for high-dimensional data and can handle non-linear relationships with the appropriate choice of kernel function. This approach is commonly used for text classification tasks due to its robustness and effectiveness.

The results with SVM model is as below:



```
Accuracy: 0.9666666666666667
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00        10
           3       1.00      1.00      1.00         7
           4       1.00      1.00      1.00         4
           5       1.00      1.00      1.00         4
           6       0.86      1.00      0.92         6
           7       1.00      0.89      0.94         9
           8       1.00      1.00      1.00         7
           9       0.83      0.83      0.83         6

    accuracy                           0.97        60
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60
```

Confusion matrix:



Compare with traditional ML models:

BERT models are preferred for complex tasks due to their ability to understand deep context and semantics, while traditional ML models are better suited for simple tasks where the feature space is well-defined and relationships are less complex. However, the choice between the two depends on various factors such as task requirements, dataset size, computational resources, and the desired level of accuracy.

Advantages and limitations:

Random Forest:

Advantages:

1. High Accuracy: Random Forest models generally achieve high accuracy in various classification tasks. They are robust to overfitting due to the aggregation of multiple decision trees.
2. Feature Importance: Random Forest models provide a feature importance score, which helps in understanding the relative importance of different features in the dataset.
3. Handles Missing Data: Random Forest models can handle missing values and maintain accuracy even when a substantial portion of the data is missing.
4. Efficient Parallelization: The training of individual decision trees in a Random Forest can be parallelized, making it computationally efficient.

Limitations:

1. Lack of Interpretability: While Random Forest provides feature importance scores, the interpretation of individual trees within the ensemble can be complex.
2. Prone to Overfitting: Despite being robust to overfitting, Random Forest models can still overfit noisy data if not properly tuned.
3. Memory Consumption: Random Forest models can consume a significant amount of memory, especially for large datasets with numerous features.
4. Black Box Nature: Like many ensemble models, Random Forest is considered a "black box" model, making it difficult to interpret the underlying decision-making process.

Support Vector Machine (SVM):

Advantages:

1. Effective in High-Dimensional Spaces: SVMs are effective in high-dimensional feature spaces, making them suitable for tasks with many features, such as text classification and image recognition.
2. Robust to Overfitting: SVMs tend to generalize well and are less prone to overfitting, especially in high-dimensional spaces.
3. Kernel Trick: SVMs can use the kernel trick to transform non-linearly separable data into a higher-dimensional space where it becomes linearly separable.
4. Global Optimum: SVMs aim to find the maximum margin hyperplane, which results in a global optimum solution.

Limitations:

1. Sensitivity to Noise: SVMs are sensitive to noise and outliers in the data, which can affect the placement of the decision boundary.
2. Computationally Intensive: Training SVM models can be computationally intensive, especially for large datasets. The time complexity of SVM training can be cubic in the number of training examples.
3. Parameter Sensitivity: SVM performance is sensitive to the choice of parameters such as the kernel function, regularization parameter (C), and kernel parameters (gamma for non-linear kernels).
4. Limited Interpretability: SVMs are often considered "black box" models, making it challenging to interpret the learned decision boundary, especially in higher-dimensional feature spaces.

• Conclusion and Future Work

Here, we aim to consider some of the reasons our bert models aren't performing perfectly:

- The most significant factor is our limited dataset. Deep learning methods are well-known for achieving optimal performance with large datasets. As data volume increases, so does the performance of these models. With our small dataset, our models may not be operating at their full potential.
- In addition to the dataset size, predicting among 10 classes poses a more challenging problem that requires more data compared to problems with fewer classes. While our best models perform well on many classes, there are some instances where performance is suboptimal. Increasing the dataset could address these issues. A 50% accuracy rate for a problem with 10 classes is considered favorable, given that random chance would result in only 10% accuracy.
- It's essential to note that poems use different vocabularies than the corpora on which the models were pretrained. This discrepancy can impact the model's performance, making it less effective in analyzing classical literature and more proficient in handling everyday language.
- Computational constraints are a consistent challenge when working with deep learning models. Due to these constraints, we were unable to utilize deeper models for our predictions.
- Overfitting occurred in our small dataset, particularly for certain poets such as Shahryar, preventing other poets from being predicted.
- Persian poems are more complex than those in other languages. Additionally, given the significant importance of rhythm in understanding the style and genre of poems, we require a model that captures this essence. Achieving this is challenging, necessitating a more complex and well-trained dataset for this purpose.

Meanwhile, classical machine learning approaches such as random forest and SVM's performance exhibited a significant improvement, with up to a 50% difference in accuracy. This can be attributed to the fact that classical machine learning methods excel when dealing with smaller datasets, whereas deep learning models typically require larger datasets to fully unleash their potential. Therefore, unless we have a substantial amount of data available, classical machine learning approaches are generally recommended for this problem.

Potential Areas for Future Improvement or Expansion:

- Incorporating machine learning algorithms to improve accuracy in author identification.
- Expanding the dataset to include a wider range of Persian authors and their works.
- Utilizing natural language processing techniques to analyze the writing styles of authors.
- Implementing deep learning models for more in-depth analysis of text features.
- Collaborating with experts in Persian literature to enhance the understanding of author styles and nuances.

Findings in this Project:

- Successful identification of authors based on their writing style in Persian literature.
- Identification of common patterns and characteristics in the works of different authors.
- Validation of the effectiveness of the methodology used for author identification.
- Insight into the unique writing styles of Persian authors and how they can be distinguished from one another.