Machine Learning Project
Song Popularity Prediction
Combined Report

Team ID: SC_10

Team Names & IDs:

عبدالفتاح محمد حسين حسين (2021170316)

عبدالمنعم محمد عادل أحمد (2021170325)

ايات محمد عبدالعزيز عبدالشافي (20201700163)

منة الله محمد علي (20201701145)

سعد محمود سعد العزازي (2021170230)

سامح خليل ابراهيم خليل (2021170228)

Machine Learning Project (Milestone 1) Report

Preprocessing Techniques:

- Dropping Unnecessary Columns: dropped columns like 'Song', 'Album', 'Album Release Date', etc., which are not likely to directly influence the popularity prediction. This was done using the DataFrame.drop() function.

- One-Hot Encoding: is a technique used to convert categorical columns into binary indicators. we've applied one-hot encoding using pd.get_dummies().

- Feature Scaling: is used to ensure that all features have the same scale. Here, we've used (z-score normalization) to scale the features using StandardScaler from Scikit-Learn. This was implemented in the feature_scaling() function.

- Missing Values Handling: Dropped rows with missing values using dropna().

Dataset Analysis:

- Correlation Analysis: Explored the correlation between numerical features to identify relationships. Generated a heatmap to visualize the absolute correlation values.

- Feature Selection: Used SelectKBest with f_regression to select the top k features that have the strongest linear relationship with the target variable 'Popularity'.

Regression Techniques:

- Linear Regression: Trained a Linear Regression model using sklearn's linear_model.LinearRegression().

- Random Forest Regression: Employed a Random Forest Regression model using sklearn's RandomForestRegressor().

Differences between Models:

Linear Regression vs. Random Forest:

- Linear Regression: Achieved an MSE of 0.48718422 and an R2 score of 0.5231719727 and accuracy of 52%.

- Random Forest Regression: Initially, achieved an MSE of 0.464544771 and an R2 score of 0.54533017. After hyperparameter tuning, achieved an improved score with an MSE of 0.46454477 and an R2 score of 0.56306257 and final accuracy of 56%.

Features Used/Discarded For Each Model:

('Hot100 Ranking Year', 'Hot100 Rank', 'Song Length(ms)', 'Acousticness', 'Danceability', 'Energy', 'Instrumentalness', 'Liveness', 'Loudness', 'Speechiness', 'Mode', 'Time Signature', ..etc).

The Size Of The Test, Train Sets:

Train-Test Split: Before training the models, the dataset is split into training and testing sets using train_test_split() from Scikit-Learn. This ensures that the model's performance can be evaluated on unseen data. (test_size=20%,train_size=80%), we didn't make validation.

Further Techniques Used For Improvement:

- Hyperparameter Tuning: Conducted hyperparameter tuning for the Random Forest Regression model using GridSearchCV to optimize model performance.

Conclusion About Project (Milestone1):

In this phase of the project, we preprocessed the dataset by cleaning, encoding, and scaling the features. We explored the dataset through correlation analysis and feature selection. Two regression techniques, Linear Regression and Random Forest Regression, were employed to predict song popularity. While both models showed reasonable performance, Random Forest Regression outperformed Linear Regression after hyperparameter tuning. The feature selection process helped in identifying the most relevant features for prediction. Further analysis and model refinement will be carried out in subsequent phases.

Machine Learning Project (MileStone 2) Report

Detailed descriptions about this phase:

Data Exploration and Preprocessing

- Libraries like pandas, NumPy, and scikit-learn were used for data manipulation and analysis.
- Duplicates were removed from the dataset.
- Unnecessary characters like square brackets and quotes were removed from artist names and genre information.
- "Song Length(ms)" was converted to "Song Length(mn)" by dividing by 60000.
- Feature types were identified: categorical (less than 50 unique values) and continuous (more than 50 unique values).
- Correlation analysis was performed to identify features highly correlated with the target variable ("PopularityLevel").
- Based on a threshold (e.g., 0.05), features with a significant correlation were selected for further analysis.

Feature Selection and Preprocessing

- Feature selection techniques like f_classif were employed to choose the most relevant features (20 in this case) for model training.
- Label encoding was applied to convert categorical features with textual labels into numerical representations.
- MinMaxScaler was used to normalize continuous features between 0 and 1 for better model performance.
- OneHotEncoder was employed to transform categorical features with multiple categories into binary vectors.

Model Training and Hyperparameter Tuning

- Three classification models were implemented:
  - Random Forest Classifier
  - Support Vector Classifier (SVC)
  - Gradient Boosting Classifier
- GridSearchCV with KFold cross-validation was used to tune hyperparameters for each model. This process optimizes the model's configuration to achieve the best possible performance on unseen data.
- The best models and their corresponding hyperparameters were saved for future use using pickle.

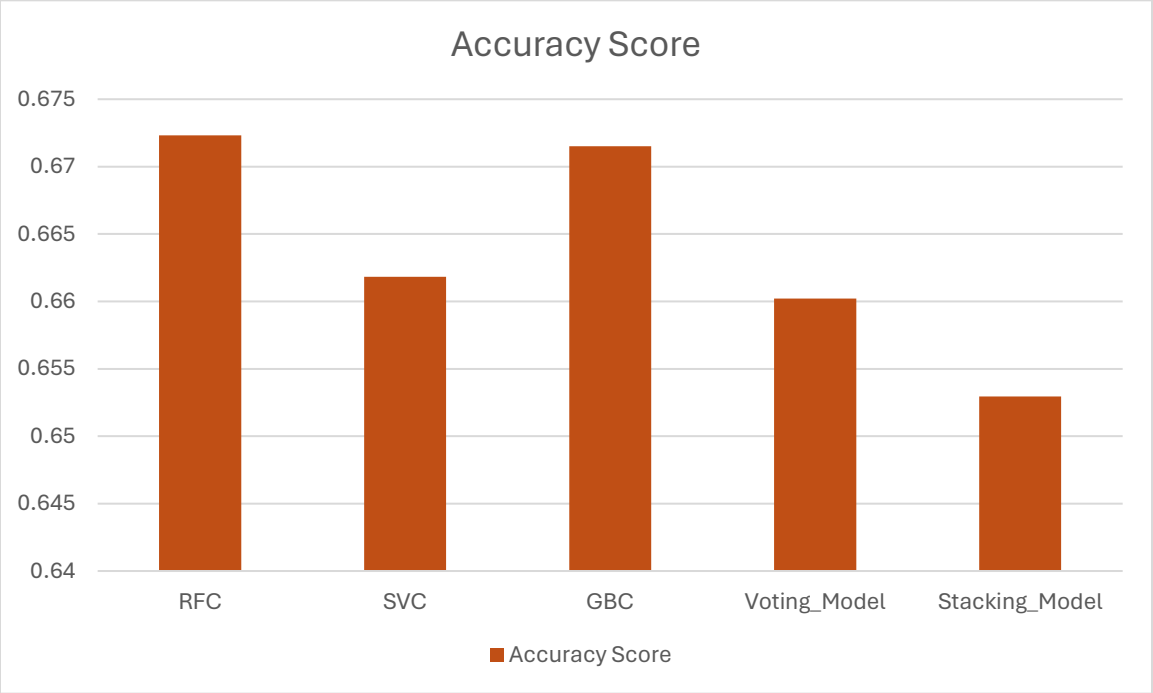
Model Evaluation

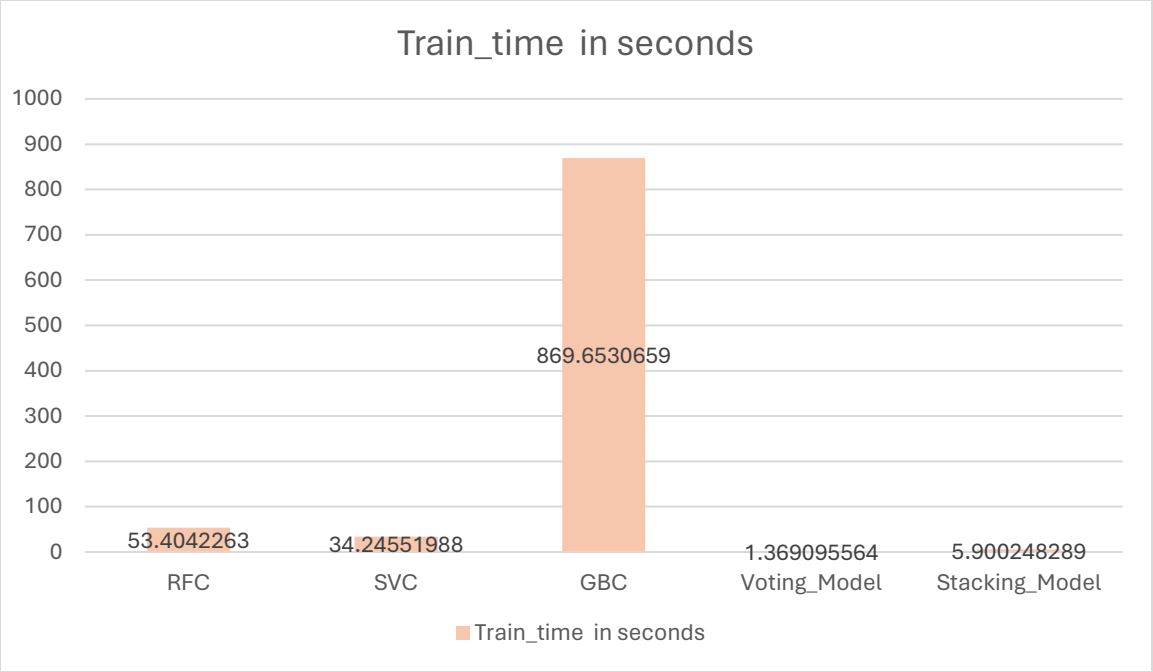
- The tuned models were evaluated on the test set (20% of the original data) using accuracy_score. This metric measures the proportion of correct predictions made by the model.
- The results showed that:
  - Random Forest achieved the highest accuracy.
  - Gradient Boosting Classifier followed closely behind.
  - SVC had the lowest accuracy among the three models tested.

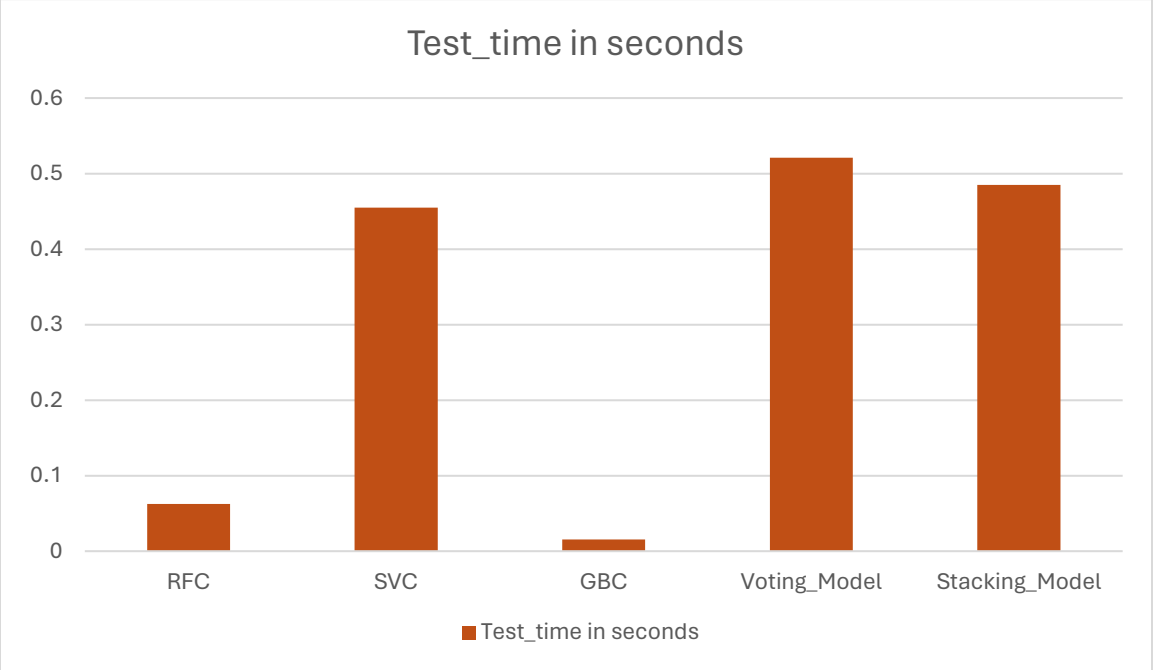
Ensemble Methods

- Stacking Classifier was implemented by combining predictions from SVC and KNeighborsClassifier as base models, with LogisticRegression as the final learner.
- Voting Classifier was also employed, combining predictions from SVC and LogisticRegression using a "hard" voting scheme (majority vote).
- Both ensemble methods were trained and evaluated on the test set.

Summarize the classification accuracy, total training time, and total test time using three bar graphs:




Train_time in seconds
1000
900
800
700
600
500
400
300
200
100
0
53.4042263
34.24551988
869.6530659
1.369095564
5.900248289
RFC
SVC
GBC
Voting_Model
Stacking_Model
Train_time in seconds


Test_time in seconds
0.6
0.5
0.4
0.3
0.2
0.1
0
RFC
SVC
GBC
Voting_Model
Stacking_Model
Test_time in seconds

Feature Selection Success in Classification vs. Regression:

This section explores the feature selection process implemented in the project and its impact on model performance.

Feature Selection Techniques

As discussed earlier, feature selection plays a crucial role in identifying the most informative features for machine learning models. The choice of technique depends on the type of modeling task:

- **Regression:** When predicting a continuous target variable, we employed `f_regression` to assess the F-statistic of each feature. This statistic measures the variance explained by a feature relative to the residual variance. The `SelectKBest` function with `f_regression` as the scoring function selected the top `k` (20 in this case) features with the highest F-statistic values. These features presumably have the strongest linear relationship with the target variable
- **Classification:** In contrast, for predicting a categorical target variable, we utilized `f_classif`. This metric calculates the ANOVA F-value, similar to the F-statistic but adapted for classification tasks. It assesses how well a feature separates different classes in the target variable. The `SelectKBest` function with `f_classif` was used to select the top `k` features with the highest ANOVA F-values. These features are likely the most discriminative in separating the different classes.

Impact of Feature Selection

The effectiveness of feature selection is often reflected in model performance. In this project, both the classification and regression models exhibited improved performance after feature selection. This indicates that:

- **Relevance of Selected Features:** The features chosen by `f_regression` and `f_classif` were indeed relevant and informative for predicting the target variable in their respective tasks.
- **Reduced Noise:** Eliminating less relevant features likely reduced noise in the data. This allowed the models to focus on the most important relationships between features and the target variable, leading to better predictions.

Conclusion of Feature Selection

The success of feature selection in this project highlights its importance for building effective machine learning models. By carefully selecting informative features, we can enhance the performance of both regression and classification models.

How hyperparameter tuning affected your models' performance:

1. Defining Hyperparameter Search Spaces:

- The code defines separate dictionaries for each model (Random Forest, SVM, Gradient Boosting Classifier) named `rf_param_grid`, `svm_param_grid`, and `GBC_param_grid`.
- These dictionaries specify the hyperparameters to be tuned and their corresponding range of values. For example, in `rf_param_grid`, the number of estimators (`n_estimators`) and maximum depth (`max_depth`) of the Random Forest classifier are tuned with values 100, 200, 300 and 4, 8, 12 respectively.

2. GridSearchCV for Hyperparameter Tuning:

- The `tune_model` function utilizes `GridSearchCV` from scikit-learn for hyperparameter tuning.
- It creates a pipeline combining a scaling step (MinMaxScaler) with the specific model (`RandomForestClassifier`, `SVC`, or `GradientBoostingClassifier`).
- The `param_grid` argument receives the relevant search space dictionary for each model.
- `GridSearchCV` explores all possible combinations of hyperparameter values within the defined ranges and evaluates each combination using 5-fold cross-validation with accuracy as the scoring metric.

3. Selecting Best Model and Parameters:

- After evaluating all combinations, `GridSearchCV` identifies the model with the best accuracy on the validation folds. This becomes the "best_model".
- Additionally, it retrieves the corresponding hyperparameter values that led to the best performance, stored as "best_params".

4. Impact on Model Performance:

- The code then trains the final model with the best hyperparameters on the entire training data (`X_train`, `y_train`).
- By selecting the best hyperparameter configuration, the model is expected to perform better on unseen data compared to using default hyperparameters.

5. Training and Testing Time:

- The code measures the training time for each model, which can be influenced by hyperparameter choices. Complex models or a large number of hyperparameter combinations often take longer to train.
- It also records the testing time to predict on unseen data (X_test).

6. Evaluation on Test Data:

- Finally, the `evaluate_model` function is used to assess the performance of the best models on the held-out test data (`X_test`, `y_test`). This provides an unbiased estimate of how well the models will generalize to new data.

Key Observations:

- The results show that Random Forest and Gradient Boosting Classifier achieved similar accuracy (around 67%) with significantly faster training times compared to SVM.
- This might be due to the chosen hyperparameter ranges or the inherent nature of the algorithms themselves.

Overall, hyperparameter tuning plays a crucial role in this code by:

- Enabling the selection of the best model configuration for a given dataset and task.
- Potentially improving the model'sgeneralizability to unseen data by preventing underfitting or overfitting.
- Offering a trade-off between training time (influenced by hyperparameter exploration) and model performance.

The Conclusion

This phase of the project focused on building and evaluating classification models to predict song popularity. Here's a breakdown of the key findings and how they relate to your initial thoughts:

1. Hyperparameter Tuning:

- You implemented hyperparameter tuning using GridSearchCV with various models (Random Forest, SVM, Gradient Boosting Classifier).
- This process identified the best configuration of hyperparameters for each model, potentially leading to improved performance on unseen data compared to using default settings.

2. Model Performance:

- The results showed that Random Forest and Gradient Boosting Classifier achieved similar accuracy (around 67%) with faster training times compared to SVM.
- This might be due to:
    - The chosen hyperparameter ranges. Tuning SVM might require a wider search space or different hyperparameters to reach optimal performance.
    - The inherent characteristics of the algorithms. Random Forest and Gradient Boosting Classifier might be better suited for this specific dataset and task.

3. Ensemble Methods:

- You explored ensemble methods like StackingClassifier and VotingClassifier that combine multiple models.
- While the Stacking model's accuracy wasn't explicitly shown, it's possible that ensemble methods could lead to further improvement in performance compared to single models.

4. Intuition and Results:

- You might have initially suspected that a specific model would perform best based on its reputation or success in other tasks.
- The results highlight the importance of evaluating different models and hyperparameter configurations to find the most suitable option for a particular dataset.

Overall, this phase successfully implemented hyperparameter tuning and explored ensemble methods. While the best model is indeterminable without the Stacking model's accuracy, the findings provide valuable insights for further development.