Model Evaluation Report Overview

In this section, we evaluate the performance of the various models based on their **Train Accuracy** and **Test Accuracy**. The models evaluated include KNN, SVC, Logistic Regression, Decision Tree, Random Forest, and the optimized Random Forest model (best at all).

Model: K-Nearest Neighbors (KNN)

Objective:

The objective was to implement the K-Nearest Neighbors (KNN) algorithm and tune the hyperparameters to optimize its performance for classification tasks.

Data Preprocessing:

The data was split into three sets:

1. Training Set (70%)
2. Validation Set (15%)
3. Test Set (15%)

Hyperparameter Tuning:

A GridSearchCV was used to identify the best hyperparameters for the KNN model. The search involved the following parameters:

- n_neighbors: Number of neighbors considered for classification (values tested: 3, 5, 7, 9).
- weights: Weight function used in prediction ('uniform' for equal weight, 'distance' for distance-based weighting).
- metric: The distance metric used to calculate the distance between data points (options tested: 'Euclidean', 'Manhattan').

❖ *Best Hyperparameters:*

n_neighbors: 9 eights: Distance metric: Manhattan

❖ *Model Training and Evaluation:*

Training Set: The model was trained on the training set using the best hyperparameters identified by GridSearchCV.

- Validation Accuracy: 0.8380
- Test Accuracy: 0.8326

The KNN model showed promising performance, achieving an accuracy of 83.8% on the validation set and 83.3% on the test set. The slight difference in performance between the validation and test sets suggests that the model generalizes well to unseen data.

Conclusion:

The K-Nearest Neighbors model with the tuned hyperparameters provided a strong performance in predicting the target variable. The model's accuracy on both the validation and test sets indicates a good fit for the data.

Model: Support Vector Classifier (SVC)

Objective:

The goal was to implement the Support Vector Classifier (SVC) and fine-tune its hyperparameters to optimize performance for the classification task.

Data Preprocessing:

The dataset was split into three sets:

1. Training Set (70%)
2. Validation Set (15%)
3. Test Set (15%)

Hyperparameter Tuning:

A GridSearchCV was used to find the best hyperparameters for the SVC model. The hyperparameters tested were:

- C: Regularization parameter controlling the trade-off between margin size and classification error (values tested: 0.1, 1).
- kernel: Kernel function used for mapping the input data into a higher-dimensional space (options tested: 'linear', 'rbf', 'poly').
- gamma: Kernel coefficient that controls the influence of individual training samples (values tested: 'scale', 'auto').

❖ Best Hyperparameters:

C: 1 kernel: RBF (Radial Basis Function) gamma: Scale

❖ Model Training and Evaluation:

Training Set: The model was trained using the best hyperparameters identified by GridSearchCV.

- ➢ Validation Accuracy: 0.8208
- ➢ Test Accuracy: 0.8171

The SVC model achieved an accuracy of 82.08% on the validation set and 81.71% on the test set, indicating strong performance on unseen data.

Conclusion:

The Support Vector Classifier with the optimized hyperparameters provided good classification performance. The model shows strong generalization capabilities, as evidenced by its similar accuracy on both the validation and test sets.

Model: Logistic Regression

Objective:

The goal was to implement the Logistic Regression model and perform hyperparameter tuning to optimize its performance for the classification task.

Data Preprocessing:

The data was split into three sets:

1. Training Set (70%)
2. Validation Set (15%)
3. Test Set (15%)

Hyperparameter Tuning:

A GridSearchCV was applied to tune the hyperparameters of the Logistic Regression model. The parameters tested were:

- C: Regularization parameter controlling the trade-off between fitting the data well and keeping the model coefficients small (values tested: 0.1, 1, 10, 100).
- penalty: Type of regularization used (values tested: 'l1', 'l2').
- solver: Algorithm to use for optimization (values tested: 'liblinear', 'saga').

❖ Best Hyperparameters:

C: 0.1 penalty: L1 (Lasso regularization) solver: SAGA (Stochastic Average Gradient Descent)

❖ Model Training and Evaluation:

Training Set: The model was trained using the best hyperparameters identified by GridSearchCV.

- Validation Accuracy: 0.7430
- Test Accuracy: 0.7338

The Logistic Regression model achieved an accuracy of 74.30% on the validation set and 73.38% on the test set, demonstrating reasonable performance, although it is slightly lower than the other models.

Conclusion:

The Logistic Regression model with L1 regularization and the SAGA solver achieved moderate performance. While it did not perform as well as the KNN and SVC models, it still provided a decent fit for the data. The model's accuracy on both the validation and test sets suggests it is useful, though improvements can be made by exploring more complex models or feature engineering.

Model: Decision Tree

Objective:

The objective was to implement a Decision Tree model and perform hyperparameter tuning to improve its classification performance.

Data Preprocessing:

The data was split into three sets:

1. Training Set (70%)
2. Validation Set (15%)
3. Test Set (15%)

Hyperparameter Tuning:

A GridSearchCV was used to identify the best hyperparameters for the Decision Tree model. The parameters tested were:

- criterion: The function to measure the quality of a split (values tested: 'gini', 'entropy').
- max_depth: The maximum depth of the tree (values tested: None, 10, 20, 30).
- min_samples_split: The minimum number of samples required to split an internal node (values tested: 2, 5, 10).
- min_samples_leaf: The minimum number of samples required to be at a leaf node (values tested: 1, 2, 4).

❖ Best Hyperparameters:

criterion: Entropy max_depth: 10 min_samples_split: 10 min_samples_leaf: 1

❖ Model Training and Evaluation:

Training Set: The model was trained using the best hyperparameters identified by GridSearchCV.

- Validation Accuracy: 0.8153
- Test Accuracy: 0.8136

The Decision Tree model achieved an accuracy of 81.53% on the validation set and 81.36% on the test set, demonstrating solid performance with a balanced split between validation and test accuracy.

Conclusion:

The Decision Tree model with entropy criterion and a maximum depth of 10 achieved a competitive performance in terms of classification accuracy. Its performance is comparable to other models, and it provides good interpretability through decision paths. The model's accuracy on both the validation and test sets indicates its effective generalization capability.

Model: Random Forest

Objective:

The goal was to implement a Random Forest model and tune its hyperparameters to optimize classification performance.

Data Preprocessing:

The data was split into three sets:

1. Training Set (70%)
2. Validation Set (15%)
3. Test Set (15%)

Hyperparameter Tuning:

A GridSearchCV was applied to find the best hyperparameters for the Random Forest model. The parameters tested were:

- n_estimators: The number of trees in the forest (values tested: 50, 100, 200).
- max_depth: The maximum depth of the trees (values tested: None, 10, 20).
- min_samples_split: The minimum number of samples required to split an internal node (values tested: 2, 5, 10).
- min_samples_leaf: The minimum number of samples required to be at a leaf node (values tested: 1, 2, 4).

Best Hyperparameters:

n_estimators: 200 max_depth: 20 min_samples_split: 2 min_samples_leaf: 1

❖ Model Training and Evaluation:

Training Set: The model was trained using the best hyperparameters identified by GridSearchCV.

- Validation Accuracy: 0.8580
- Test Accuracy: 0.8609

The Random Forest model achieved an accuracy of 85.80% on the validation set and 86.09% on the test set, demonstrating excellent performance and strong generalization capability.

Conclusion:

The Random Forest model, with 200 estimators and a maximum depth of 20, delivered the highest accuracy among all tested models. Its performance on both the validation and test sets shows its ability to generalize well and perform robustly across different subsets of data. The model's ensemble nature contributed to its high performance, indicating its effectiveness for the classification task.

Model: Random Forest (Optimized)

Objective:

The goal was to implement a highly optimized Random Forest model using selected hyperparameters to achieve the best classification performance.

Data Preprocessing:

The data was split into three sets:

1. Training Set (70%)
2. Validation Set (15%)
3. Test Set (15%)

Hyperparameters:

The final model was configured with the following hyperparameters:

- n_estimators: 300 (Number of trees in the forest)
- max_depth: 12 (Limit depth to avoid overfitting)
- min_samples_split: 17 (Avoid very small splits)
- min_samples_leaf: 2 (Avoid deep leaves to prevent overfitting)
- max_features: 'sqrt' (Good default for Random Forest)
- criterion: 'entropy' (Used for information gain)
- bootstrap: False (Trees are built on the entire dataset without bootstrapping)
- random_state: 0 (For reproducibility)

Model Training and Evaluation:

- Training Set Accuracy: 90.55%
- Test Set Accuracy: 85.52%

The model achieved an accuracy of 90.55% on the training set and 85.52% on the test set. The performance on the test set is strong, with a small gap compared to the training set, indicating that the model has generalization capability without overfitting.

Conclusion:

The Random Forest model with optimized hyperparameters performed excellently, achieving the highest accuracy on the training set and very strong performance on the test set. The slight reduction in accuracy on the test set compared to the training set suggests the model is well-regularized and generalizes well to unseen data. This model is the best-performing model overall.

Model Performance Comparison

In this section, we evaluate the performance of the various models based on their Train Accuracy and Test Accuracy. The models evaluated include KNN, SVC, Logistic Regression, Decision Tree, Random Forest, and the optimized Random Forest model (bestatall).

Model	Train Accuracy	Test Accuracy
KNN	99.99%	83.26%
SVC	83.16%	81.71%
Logistic Regression	75.67%	73.38%
Decision Tree	86.21%	81.36%
Random Forest	99.99%	86.09%
bestatall (Optimized Random Forest)	90.55%	85.52%

Key Observations:

- KNN and Random Forest achieved perfect accuracy on the training set (99.99%), showing a strong fit to the training data. However, KNN exhibited a significant drop in test accuracy (83.26%), indicating potential overfitting.
- SVC and Logistic Regression showed moderate performance, with SVC performing better than Logistic Regression both on training and test sets.
- Decision Tree performed well, with a balance between training and test accuracy.
- Random Forest showed excellent generalization with a high-test accuracy (86.09%).

❖ The bestatall (Optimized Random Forest) model performed well, with 90.55% on the training set and 85.52% on the test set, making it the best optimized model overall.