ML Project

Aaditya Gole, Daksh Rajesh, Harshavarshan R

December 12, 2024

Project Link

Contents

1 Introduction

In this project, we applied various machine learning models to a dataset to predict the target variable. The models used include K-Nearest Neighbors (KNN), Logistic Regression, XGBoost with Grid Search, Support Vector Machine (SVM), and Neural Networks. We performed data preprocessing, hyperparameter tuning, and evaluated the models based on their accuracy.

2 Data Preprocessing

The dataset required several preprocessing steps to ensure optimal performance of the models. The preprocessing steps included:

- **Data Cleaning**:
  - Loaded the dataset and dropped any rows with missing values (NaNs) to maintain data integrity.
- **Feature Selection**:
  - Selected only numeric columns for model training to simplify the analysis.
- **Outlier Detection and Handling**:
  - Used boxplots to visualize and detect outliers in features such as Age, Income, Loan Amount, etc.
  - Addressed outliers by capping/extending them to reduce their impact on the models.
- **Feature Scaling**:
  - Standardized the features using `StandardScaler` to ensure all features have a mean of 0 and a standard deviation of 1.
- **Data Splitting**:
  - Split the data into training and test sets, typically with an 80-20 split, to evaluate model performance on unseen data.

3 Models and Hyperparameter Tuning

3.1 K-Nearest Neighbors (KNN)

We implemented the KNN algorithm to classify the data points based on their nearest neighbors. Key steps include:

- **Parameter Selection**:

- Chose the number of neighbors (k) empirically, testing values from 1 to 15.
- Selected the k value that resulted in the highest cross-validation accuracy.

- **Training**:
  - Used the standardized features for training to ensure fair distance calculations.
  - Trained the model on the entire training dataset.

- **Evaluation**:
  - Evaluated the model on the test set to assess its generalization capability.

The accuracy achieved by the KNN model is 0.87325.

3.2 Logistic Regression

For logistic regression:

- **Feature Scaling**:
  - Performed standardization to ensure features are on the same scale.

- **Hyperparameter Tuning**:
  - Used grid search for hyperparameter tuning, exploring different values of the regularization parameter C and solver methods ('liblinear', 'saga').
  - Implemented 5-fold cross-validation to find the best hyperparameters.

- **Training**:
  - Trained the logistic regression model with the best-found hyperparameters on the training set.

- **Evaluation**:
  - Evaluated the model on the validation set to check for overfitting and generalization.

The accuracy achieved by the Logistic Regression model is 0.88500.

3.3 XGBoost with Grid Search

The XGBoost model was implemented with extensive hyperparameter tuning:

- **Hyperparameter Tuning**:
  - Used grid search to find the optimal combination of parameters such as `n_estimators`, `max_depth`, `learning_rate`, and `subsample`.
  - Performed cross-validation to prevent overfitting and select the best model.
- **Training**:
  - Trained the XGBoost model on the training set using the best hyperparameters.
- **Feature Importance**:
  - Analyzed feature importance scores provided by the model to understand influential predictors.

The accuracy achieved by the XGBoost model is 0.88713.

3.4 Support Vector Machine (SVM)

For the SVM model:

- **Kernel Selection**:
  - Implemented both linear and nonlinear kernels (e.g., RBF kernel).
  - Selected the best kernel based on cross-validation accuracy.
- **Hyperparameter Tuning**:
  - Tuned parameters such as the regularization parameter C and kernel coefficient γ using grid search.
  - Used cross-validation to find the optimal combination of hyperparameters.
- **Training and Evaluation**:
  - Trained the SVM model on the training set with standardized features.
  - Evaluated the model on the test set to assess performance.

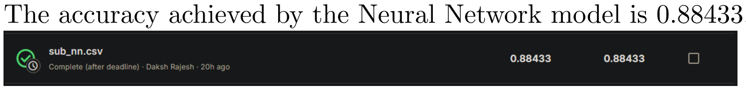
The accuracy achieved by the SVM model is 0.88477.

3.5 Neural Networks

In the neural network implementation:

- **Model Architecture**:
  - Built a multi-layer perceptron using TensorFlow/Keras with input, hidden, and output layers.
  - Experimented with different network architectures by varying the number of layers and neurons.
- **Activation Functions**:
  - Used ReLU activation functions in hidden layers for non-linearity.
  - Applied sigmoid activation in the output layer for binary classification.
- **Regularization Techniques**:
  - Implemented dropout layers to prevent overfitting by randomly deactivating neurons during training.
  - Used L2 regularization on weights to penalize large weights.
- **Hyperparameter Tuning**:
  - Adjusted learning rate, batch size, and the number of epochs.
  - Used early stopping based on validation loss to prevent overfitting.
- **Training and Evaluation**:
  - Compiled the model using the Adam optimizer and binary cross-entropy loss function.
  - Trained the model on the training set with validation splits to monitor performance.

The accuracy achieved by the Neural Network model is 0.88433.



4 Results and Comparison

Model	Accuracy
K-Nearest Neighbors	0.87325
Logistic Regression	0.88500
XGBoost	0.88713
Support Vector Machine	0.88477
Neural Networks	0.88433

Table 1: Comparison of model accuracies

5 Conclusion

In this project, we explored various machine learning models and their performance on the dataset. Each model underwent thorough preprocessing and hyperparameter tuning to optimize results. The comparison shows that:

- **XGBoost** achieved the highest accuracy of 0.88713, indicating its effectiveness in handling complex data patterns.
- **Logistic Regression** and **SVM** also performed well, suggesting that the dataset has features that are linearly separable.
- **Neural Networks** provided competitive results but did not surpass XGBoost, possibly due to limited data or the need for more complex architectures.
- **KNN** showed the lowest accuracy, which may be attributed to its sensitivity to the value of k and the curse of dimensionality.

Overall, ensemble methods like XGBoost proved to be the most effective for this dataset. Future work could explore:

- **Feature Engineering**: Creating new features or transforming existing ones to improve model performance.
- **Advanced Models**: Investigating other ensemble methods or deep learning architectures.
- **Cross-Validation**: Employing more robust cross-validation techniques to ensure model generalization.
- **Data Augmentation**: Increasing the dataset size to enhance the training of models like Neural Networks.