Project Report

Title: Predicting Student Results Using Logistic Regression and Neural Networks

Abstract

This project aims to predict student results based on various performance metrics using Logistic Regression and Neural Network models. The dataset includes marks in Python, Stats and ML, and SQL, along with their respective feedback. The project includes data preprocessing, feature engineering, feature selection, model training, and evaluation. Visualisation techniques are also employed to gain insights from the data.

Introduction

The dataset contains evaluation metrics for students, which include marks and feedback in different subjects. The primary goal is to predict the 'Result' of the students, which indicates whether a student has passed or failed based on their performance metrics.

Data Preprocessing

1. **Loading the Dataset:**
2. **Dropping Unnecessary Columns:** Unnecessary columns such as '**Email'**, '**Python Feedback**', and '**SQL Feedback**' are removed to focus on relevant features.

3. **Encoding Categorical 'Result' Column:** The '**Result**' column, which is categorical, is encoded into numerical values using Label Encoding.
4. **Outlier Removal Using IQR:** Outliers are removed based on the Interquartile Range (IQR) method to ensure data quality and reliability.
5. **Feature Engineering:** A new feature '**Total %**' is created by averaging the percentages of Python, Stats and ML, and SQL marks to capture overall student performance.

Feature Scaling and Selection

1. **Feature Scaling:** The feature values are standardized to have zero mean and unit variance using StandardScaler to ensure that all features contribute equally to the model.
2. **Feature Selection:** Recursive Feature Elimination (RFE) with Logistic Regression is used to select the top 5 features.

Logistic Regression Model

Training:

logistic_model = LogisticRegression(random_state=42)

logistic_model.fit(X_train, y_train)

Evaluation:

y_pred_logistic = logistic_model.predict(X_test)

report_logistic = classification_report(y_test, y_pred_logistic)

print(report_logistic)

Neural Network Model

Building and Compiling:

```
nn_model = Sequential()
nn_model.add(Dense(64, input_dim=X_selected.shape[1], activation='relu'))
nn_model.add(Dense(32, activation='relu'))
nn_model.add(Dense(2, activation='softmax'))
nn_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

Training:

```
nn_model.fit(X_train, y_train_nn, epochs=50, batch_size=8, verbose=0,
validation_split=0.1)
```

Evaluation:

```
loss, accuracy = nn_model.evaluate(X_test, y_test_nn, verbose=0)
y_pred_nn = nn_model.predict(X_test)
y_pred_nn_classes = y_pred_nn.argmax(axis=1)
report_nn = classification_report(y_test, y_pred_nn_classes)
print(report_nn)
```

Visualization and Insights

1. **Distribution of Marks:** Histograms showing the distribution of marks in Python, Stats and ML, and SQL.
2. **Correlation Heatmap:** Heatmap showing the correlation between different features in the dataset.
3. **Confusion Matrix:** Confusion matrices for both Logistic Regression and Neural Network models to visualize the performance in terms of correctly and incorrectly classified instances.
4. **Feature Importance:** Bar plot showing the importance of the selected features as determined by the Logistic Regression model.

Results

- **Logistic Regression Model:**
  - Precision, recall, and F1-score for each class are displayed.
  - Confusion matrix visualization.
- **Neural Network Model:**
  - Accuracy of the model on the test set.
  - Precision, recall, and F1-score for each class are displayed.
  - Confusion matrix visualization.

Conclusion

Both Logistic Regression and Neural Network models were successfully trained to predict the 'Result' of students. The project demonstrates the importance of data preprocessing, feature engineering, and feature selection in building effective machine learning models. Visualization techniques provided valuable insights into the data distribution, feature importance, and model performance.

Future Work

Future work could include exploring more advanced feature engineering techniques, trying different machine learning algorithms, and tuning hyperparameters to further improve model performance. Additionally, incorporating more data and using cross-validation techniques could help in building a more robust model.