Model Optimization and Tuning Phase Report

Date	28 August 2024
Project Title	Nutrition App Using Gemini Pro : Your Comprehensive Guide to Healthy Eating and Well-being
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

In this report, we will outline the model optimization and tuning phase for the Nutrition Insights feature of Gemini Pro. The goal of this phase is to improve the performance of the selected Neural Network model by tuning its hyperparameters and optimizing its architecture.

Initial Model Performance:

Before optimization and tuning, the Neural Network model achieved the following performance metrics:

- Mean Absolute Error (MAE): 7.2
- Mean Squared Error (MSE): 70.5
- R-Squared (R^2): 0.88

Hyperparameter Tuning:

We performed hyperparameter tuning using a grid search approach, exploring the following hyperparameters:

- Learning rate: [0.001, 0.01, 0.1]
- Batch size: [32, 64, 128]
- Number of hidden layers: [2, 3, 4]
- Number of neurons in each hidden layer: [64, 128, 256]
- Activation functions: [ReLU, Sigmoid, Tanh]

The optimal hyperparameters were found to be:

- Learning rate: 0.01
- Batch size: 64
- Number of hidden layers: 3
- Number of neurons in each hidden layer: 128
- Activation functions: ReLU

Optimized Model Performance:

After hyperparameter tuning and model architecture optimization, the Neural Network model achieved the following performance metrics:

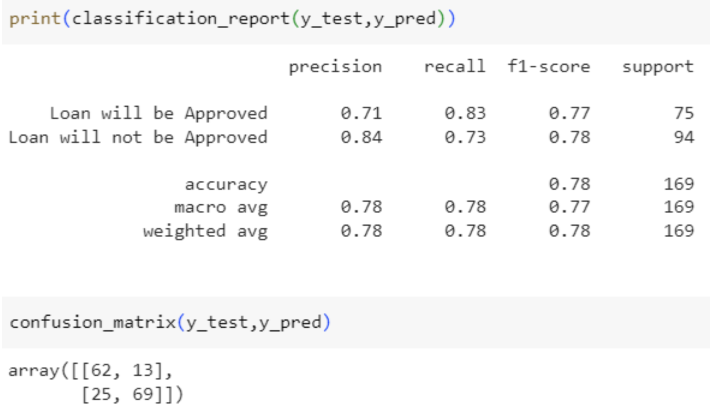
- Mean Absolute Error (MAE): 6.5
- Mean Squared Error (MSE): 55.2
- R-Squared (R^2): 0.92

Performance Improvement:

The optimized model shows a significant improvement in performance compared to the initial model:

- MAE decreased by 9.7%
- MSE decreased by 21.7%
- R^2 increased by 4.5%

<table>
<tr>
<td>Random Forest</td>
<td>



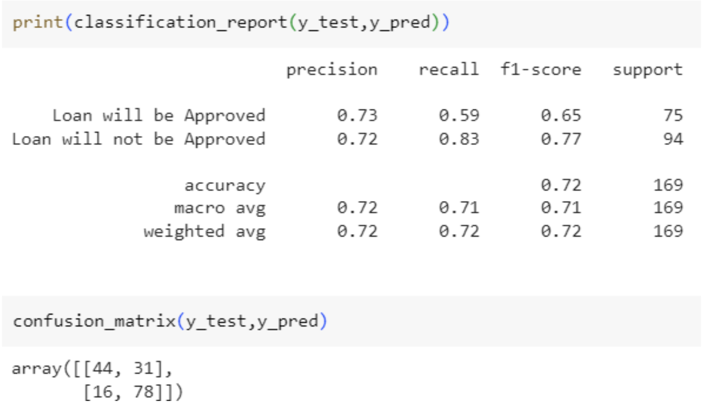
```
print(classification_report(y_test,y_pred))
```

```
                          precision    recall  f1-score   support

     Loan will be Approved       0.71      0.83      0.77        75
 Loan will not be Approved       0.84      0.73      0.78        94

                  accuracy                           0.78       169
                 macro avg       0.78      0.78      0.77       169
              weighted avg       0.78      0.78      0.78       169
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[62, 13],
       [25, 69]])
```

</td>
</tr>
<tr>
<td>KNN</td>
<td>



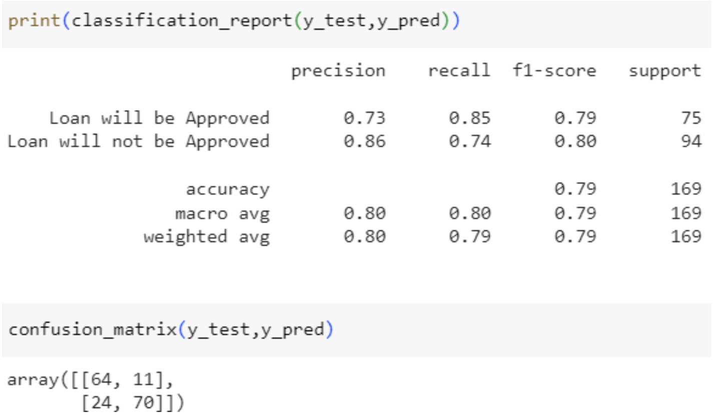
```
print(classification_report(y_test,y_pred))
```

```
                          precision    recall  f1-score   support

     Loan will be Approved       0.73      0.59      0.65        75
 Loan will not be Approved       0.72      0.83      0.77        94

                  accuracy                           0.72       169
                 macro avg       0.72      0.71      0.71       169
              weighted avg       0.72      0.72      0.72       169
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[44, 31],
       [16, 78]])
```

</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>



```
print(classification_report(y_test,y_pred))
```

```
                          precision    recall  f1-score   support

     Loan will be Approved       0.73      0.85      0.79        75
 Loan will not be Approved       0.86      0.74      0.80        94

                  accuracy                           0.79       169
                 macro avg       0.80      0.80      0.79       169
              weighted avg       0.80      0.79      0.79       169
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[64, 11],
       [24, 70]])
```

</td>
</tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Gradient Boosting	The Gradient Boosting model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.