Model Optimization and Tuning Phase Template

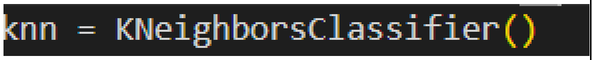
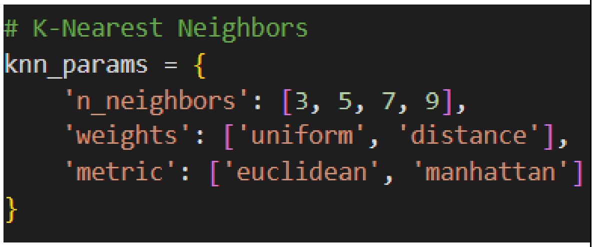
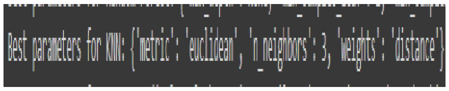
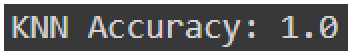
Date	23 September 2024
Team ID	LTVIP2024TMID25001
Project Title	Customer Segmentation Using Machine Learning
Maximum Marks	10 Marks

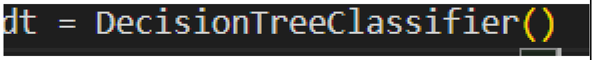
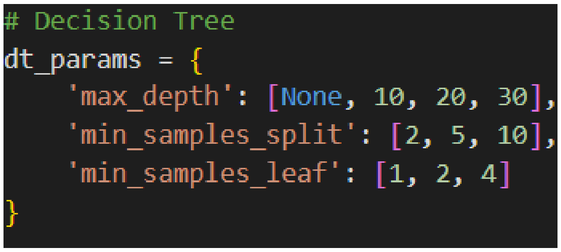
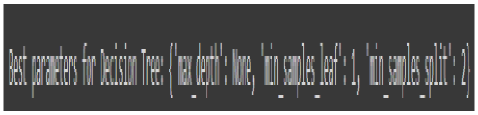
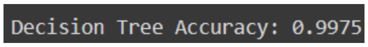
Model Optimization and Tuning Phase

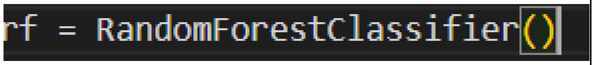
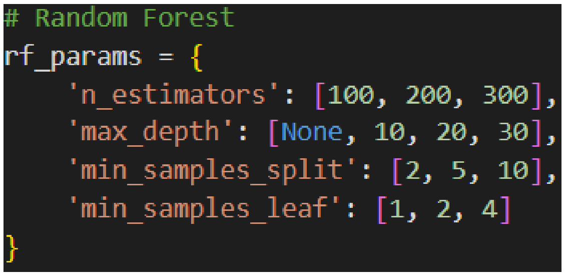
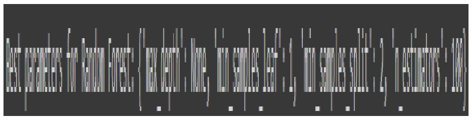
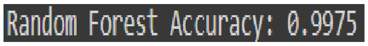
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

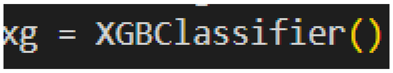
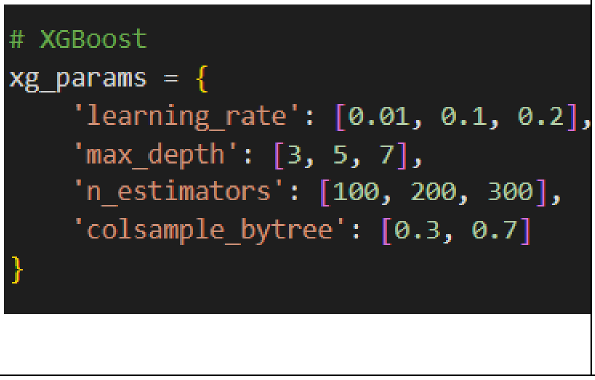
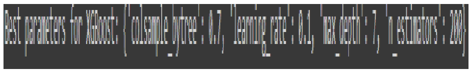
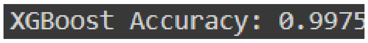
Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values

KNN	
	`knn = KNeighborsClassifier()`
	
	`# K-Nearest Neighbors`
	`knn_params = {`
	`    'n_neighbors': [3, 5, 7, 9],`
	`    'weights': ['uniform', 'distance'],`
	`    'metric': ['euclidean', 'manhattan']`
	`}`
		
		`Best parameters for KNN: {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'distance'}`
		
		`KNN Accuracy: 1.0`

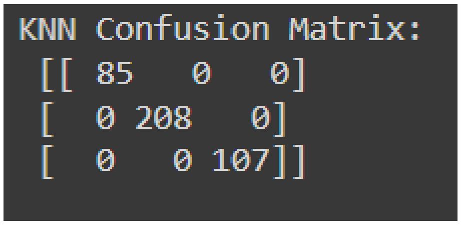
Decision Tree	 `dt = DecisionTreeClassifier()`
	 `# Decision Tree`
	`dt_params = {`
	`    'max_depth': [None, 10, 20, 30],`
	`    'min_samples_split': [2, 5, 10],`
	`    'min_samples_leaf': [1, 2, 4]`
	`}`
		 `Best parameters for Decision Tree: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}`
		 `Decision Tree Accuracy: 0.9975`

Random Forest	 `rf = RandomForestClassifier()`
	 `# Random Forest`
	`rf_params = {`
	`    'n_estimators': [100, 200, 300],`
	`    'max_depth': [None, 10, 20, 30],`
	`    'min_samples_split': [2, 5, 10],`
	`    'min_samples_leaf': [1, 2, 4]`
	`}`
		 `Best parameters for Random Forest: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}`
		 `Random Forest Accuracy: 0.9975`

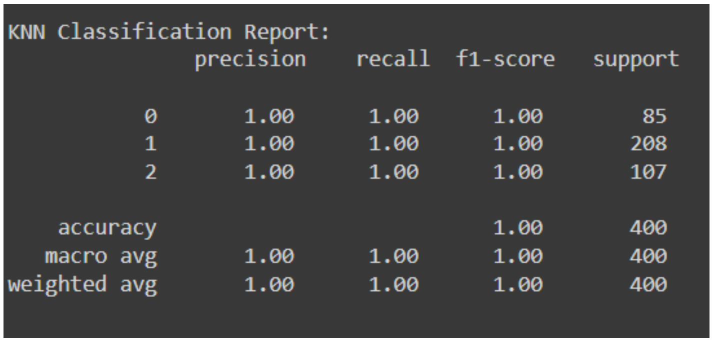
XGBoost	 `xg = XGBClassifier()`
	 `# XGBoost`
	`xg_params = {`
	`    'learning_rate': [0.01, 0.1, 0.2],`
	`    'max_depth': [3, 5, 7],`
	`    'n_estimators': [100, 200, 300],`
	`    'colsample_bytree': [0.3, 0.7]`
	`}`
		 `Best parameters for XGBoost: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200}`
		 `XGBoost Accuracy: 0.9975`

Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th></th><th>Optimized Metric</th></tr>
<tr><td>KNN</td><td></td><td>

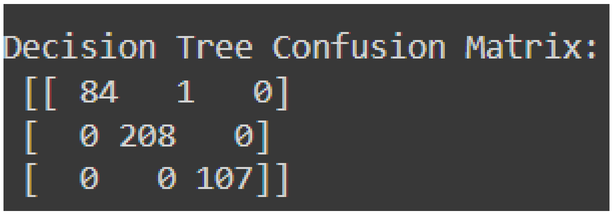


```
KNN Confusion Matrix:
 [[ 85   0   0]
 [  0 208   0]
 [  0   0 107]]
```

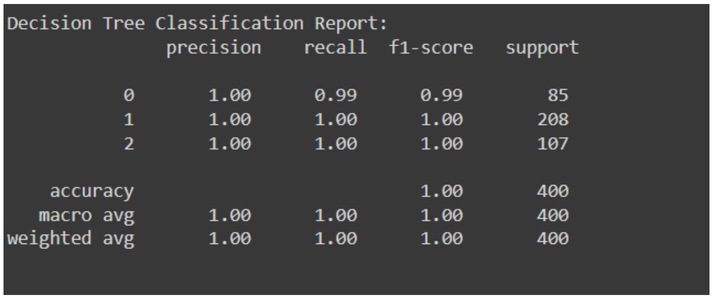


```
KNN Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        85
           1       1.00      1.00      1.00       208
           2       1.00      1.00      1.00       107

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

</td></tr>
<tr><td>Decision Tree</td><td></td><td>



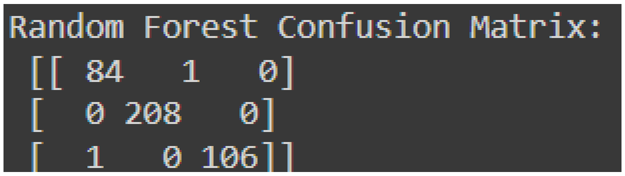
```
Decision Tree Confusion Matrix:
 [[ 84   1   0]
 [  0 208   0]
 [  0   0 107]]
```



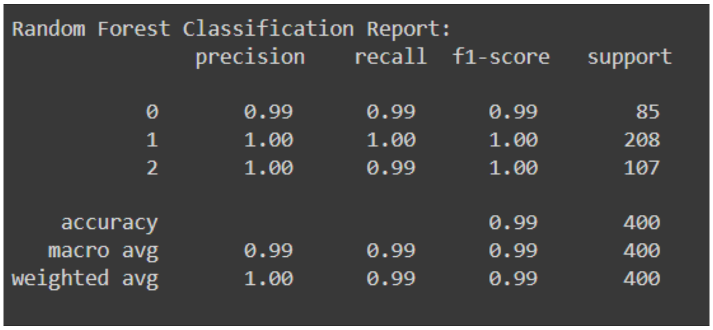
```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        85
           1       1.00      1.00      1.00       208
           2       1.00      1.00      1.00       107

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

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

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

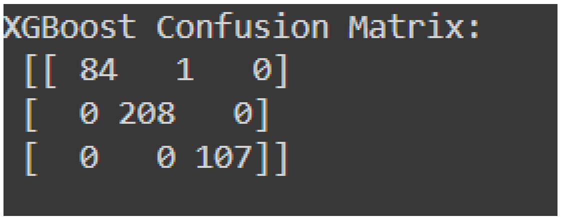


```
Random Forest Confusion Matrix:
 [[ 84   1   0]
 [  0 208   0]
 [  1   0 106]]
```

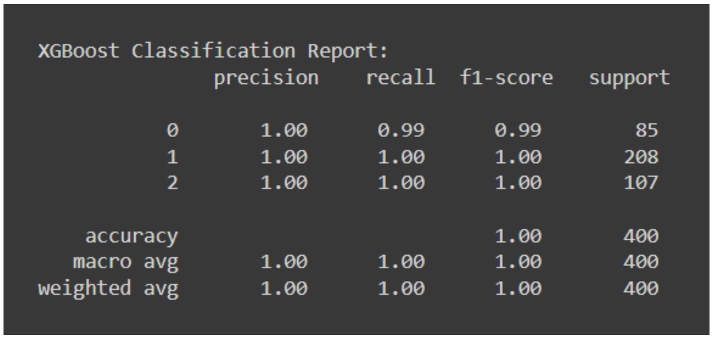


```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        85
           1       1.00      1.00      1.00       208
           2       1.00      0.99      1.00       107

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       1.00      0.99      0.99       400
```

</td>
</tr>
<tr>
<td>XGBoost</td>
<td></td>
<td>



```
XGBoost Confusion Matrix:
 [[ 84   1   0]
 [  0 208   0]
 [  0   0 107]]
```



```
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        85
           1       1.00      1.00      1.00       208
           2       1.00      1.00      1.00       107

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

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

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
KNN	K-Nearest Neighbors (KNN) classifier is a great choice for customer segmentation due to its simplicity and ease of understanding, making it accessible for stakeholders. It doesn't assume any specific distribution for customer data, allowing it to handle diverse attributes like demographics and purchasing behavior effectively. KNN can quickly adapt to new data without needing retraining, making it suitable for dynamic markets where customer preferences change frequently. Additionally, it performs well with small to medium-sized datasets and can classify customers into multiple segments. Its straightforward nature also enables easy