Model Optimization and Tuning Phase Report

Date	23 September 2024
Team ID	LTVIP2024TMID24997
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
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

Linear Regression	import numpy as np
	from sklearn.linear_model import LinearRegression
	from sklearn.metrics import mean_squared_error, r2_score
	
	# Model training
	linear_reg = LinearRegression()
	linear_reg.fit(x_train, y_train)
	
	# Prediction
	linear_pred = linear_reg.predict(x_test)
	
	# Evaluation
	print("MSE:", mean_squared_error(y_test, linear_pred))
	print("R2 Score:", r2_score(y_test, linear_pred))
	
	# Custom accuracy score based on a tolerance level
	def accuracy_score(y_true, y_pred, tolerance=0.1):
	    return np.mean(np.abs(y_true - y_pred) <= tolerance)
		# Evaluation
		print("MSE:", mean_squared_error(y_test, linear_pred))
		print("R² Score:", r2_score(y_test, linear_pred))
		
		# Calculate accuracy using R² score
		accuracy = r2_score(y_test, linear_pred)
		print("R² Score (Accuracy):", accuracy)
		
		MSE: 0.011427751339300353
		R² Score: 0.9999997047756003
		R² Score (Accuracy): 0.9999997047756003

Ridge regression	def accuracy_score(y_true, y_pred, tolerance=0.1):
	    return np.mean(np.abs(y_true - y_pred) <= tolerance)
	
	# Standardize features
	scaler = StandardScaler()
	
	# Create a pipeline with scaling and Ridge regression
	ridge_pipeline = Pipeline([
	    ('scaler', scaler),
	    ('ridge', Ridge())
	])
	
	# Hyperparameter tuning using GridSearchCV
	param_grid = {
	    'ridge__alpha': [0.01, 0.1, 1.0, 10.0, 100.0] # Range of alpha value
	}
	
	grid_search = GridSearchCV(ridge_pipeline, param_grid, cv=5, scoring='r2'
	grid_search.fit(x_train, y_train)
		# Evaluation
		print("MSE:", mean_squared_error(y_test, ridge_pred))
		print("R² Score:", r2_score(y_test, ridge_pred))
		
		# Calculate accuracy using R² score
		accuracy = r2_score(y_test, ridge_pred)
		print("R² Score (Accuracy):", accuracy)
		
		MSE: 0.0720158246881738
		R² Score: 0.99999813954399939
		R² Score (Accuracy): 0.9999981395439939

Lasso regression

```python
# Define a custom accuracy score based on a tolerance level
def accuracy_score(y_true, y_pred, tolerance=0.1):
    return np.mean(np.abs(y_true - y_pred) <= tolerance)

# Standardize features
scaler = StandardScaler()

# Create a pipeline with scaling and Lasso regression
lasso_pipeline = Pipeline([
    ('scaler', scaler),
    ('lasso', Lasso())
])

# Hyperparameter tuning using GridSearchCV
param_grid = {
    'lasso__alpha': [0.01, 0.1, 1.0, 10.0, 100.0]  # Range of alpha values to test
}

grid_search = GridSearchCV(lasso_pipeline, param_grid, cv=5, scoring='r2')  # Cross-validation
grid_search.fit(x_train, y_train)
```

```python
# Calculate accuracy using R² score
accuracy = r2_score(y_test, lasso_pred)
print("R² Score (Accuracy):", accuracy)
print("Best Parameters:", grid_search.best_params_)
```

```
MSE: 0.012321835829863403
R² Score: 0.9999996816778316
R² Score (Accuracy): 0.9999996816778316
Best Parameters: {'lasso__alpha': 0.01}
```

Decision Tree

```python
# Define hyperparameter grid for Decision Tree Regression
dt_param_dist = {
    'dt__max_depth': [None, 3, 5, 10, 15],          # Maximum de
    'dt__min_samples_split': [2, 5, 10],           # Minimum sam
    'dt__min_samples_leaf': [1, 2, 4, 6],          # Minimum sam
    'dt__max_features': ['auto', 'sqrt', 'log2']   # Number of f
}

# Use RandomizedSearchCV for hyperparameter tuning
dt_search = RandomizedSearchCV(
    dt_pipeline,
    dt_param_dist,
    n_iter=50,
    cv=5,
    scoring='r2',
    n_jobs=-1,
    random_state=42
)

# Fit the model
dt_search.fit(X_train, y_train)
```

```python
best_dt = grid_search_dt.best_estimator_
print("Best Hyperparameters for Decision Tree:", grid_search_dt.best_para
y_pred_dt = best_dt.predict(X_test)
print("Decision Tree Accuracy: {:.2f}".format(accuracy_score(y_test, y_pr
```

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best Hyperparameters for Decision Tree: {'criterion': 'gini', 'max_depth'
Decision Tree Accuracy: 0.74
```

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Linear Regression	R2: 0.999	R2: 0.999
Ridge Regression	R2: 0.998	R2: 0.998
Lasso Regression	R2: 0.996	R2: 0.996
Decision Tree Regressor	R2: 0.7445	R2: 0.7445

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Linear Regression	The Linear Regression model was chosen as the final optimized model because it exhibited the highest R-squared value (0.999), indicating a strong fit to the data. Additionally, it had a lower MSE (0.0114) and Accuracy (99%) compared to other models, suggesting superior predictive accuracy.