FlexSaf Internship Final Project: Machine Learning Model Training with Scikit-learn

Complete Project Documentation

Executive Summary

This project demonstrates the implementation of multiple machine learning regression models to predict shipping times based on shipment characteristics. Using Python and scikit-learn, we developed and compared five different algorithms to identify the most effective approach for shipping time prediction.

Key Results:

- **Best Model**: LightGBM with R^2 = 0.9264 and MSE = 0.2533
- **Worst Model**: Linear Regression with R^2 = 0.7794 and MSE = 0.7589
- **Dataset**: 25,000 shipment records with 6 features
- **Prediction Accuracy**: Up to 92.6% variance explained

---

1. Project Objective

Primary Goal: Create a simple machine learning model to predict shipping times (outcomes) based on shipment features using Python and scikit-learn.

Business Value: Accurate shipping time predictions enable better customer expectations, logistics planning, and operational efficiency.

---

2. Methodology

2.1 Applied Methods

1. **Feature Engineering -** Categorical variable encoding
2. **Data Preprocessing -** Train/test split and feature scaling
3. **Model Training -** Multiple regression algorithms
4. **Performance Evaluation -** MSE and R^2 metrics comparison
5. **Prediction Testing -** Sample predictions on new data

2.2 Technical Stack

- **Python 3.x -** Primary programming language

- **pandas** - Data manipulation and analysis
- **scikit-learn** - Machine learning algorithms and preprocessing
- **matplotlib/seaborn** - Data visualization
- **numpy** - Numerical operations
- **LightGBM** - Gradient boosting implementation
- **XGBoost** - Extreme gradient boosting

---

3. Dataset Description

3.1 Data Overview

- **Total Records**: 25,000 shipments
- **Features**: 6 variables (3 numerical, 3 categorical)
- **Target Variable**: shipping_times (days)
- **Data Quality**: Clean dataset with no missing values

3.2 Feature Specifications

Feature	Type	Description	Example Values
routes	Categorical	Shipping route identifier	Route A, Route B, Route C
shipping_times	Numerical (Target)	Delivery time in days	3, 4, 7
distance(km)	Numerical	Distance in kilometers	967, 1011, 1058
number_of_products_sold	Numerical	Product quantity	177, 254, 510
transportation_modes	Categorical	Transport method	Air, Rail, Road, Sea
location	Categorical	Destination city	Mumbai, Chennai, Delhi, Kolkata

3.3 Sample Data

	routes	shipping_times	distance(km)	number_of_products_sold	transportation_modes	location
0	Route B	3	1016	254	Air	Mumbai
1	Route C	4	967	510	Rail	Mumbai
2	Route A	3	1011	177	Air	Mumbai
3	Route A	4	1039	195	Rail	Chennai
4	Route B	7	1058	281	Sea	Kolkata

---

4. Data Preprocessing

4.1 Feature Engineering

One-Hot Encoding Implementation:

- Transformed categorical variables into binary columns
- Applied to: `transportation_modes`, `location`, `routes`
- Used `drop_first=True` to avoid multicollinearity
- Final feature count: 11 features

Encoded Features:

```
Original Features: ['routes', 'distance(km)', 'number_of_products_sold',
                    'transportation_modes', 'location']

Encoded Features: ['distance(km)', 'number_of_products_sold',
                   'transportation_modes_Rail', 'transportation_modes_Road',
                   'transportation_modes_Sea', 'location_Chennai',
                   'location_Delhi', 'location_Kolkata', 'location_Mumbai',
                   'routes_Route B', 'routes_Route C']
```

4.2 Feature Scaling

StandardScaler Application:

- Normalized numerical features to zero mean, unit variance
- Applied to: `distance(km)`, `number_of_products_sold`
- Ensures equal contribution from all features to model training

Scaling Results:

- Mean ≈ 0 for all scaled features
- Standard deviation = 1.0 for all scaled features

4.3 Train-Test Split

- **Training Set**: 18,750 samples (75%)
- **Test Set**: 6,250 samples (25%)
- **Random State**: 30 (for reproducibility)
- **Stratification**: Not applied (regression problem)

---

5. Model Implementation

5.1 Model Selection Strategy

Selected diverse algorithms to capture different relationship patterns:

1. **Linear Regression** - Baseline linear relationship model
2. **Random Forest** - Ensemble method for non-linear patterns
3. **Support Vector Regression (SVR)** - Kernel-based non-linear modeling
4. **LightGBM** - Efficient gradient boosting implementation
5. **XGBoost** - Advanced gradient boosting with regularization

5.2 Model Configurations

Linear Regression

```python
LinearRegression()
# Default parameters - no hyperparameter tuning required
```

Random Forest Regressor

```python
RandomForestRegressor(random_state=32)
# Uses default parameters: n_estimators=100, max_features='sqrt'
```

Support Vector Regression

```python
SVR(kernel='rbf', C=10, epsilon=0.1)
# RBF kernel for non-linear relationships
# C=10: Regularization parameter
# epsilon=0.1: Tolerance for error
```

LightGBM Regressor

```python
```

```
LGBMRegressor(n_estimators=5, learning_rate=0.6, max_depth=7, random_state=32)
# Fast training with fewer estimators
# Higher learning rate for quick convergence
# Moderate depth to prevent overfitting
```

XGBoost Regressor

```
python

XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=5, random_state=32)
# Standard configuration
# Conservative learning rate
# Moderate depth for generalization
```

---

6. Results and Performance Analysis

6.1 Model Performance Comparison

Model	MSE	R^2 Score	Rank
LightGBM	**0.2533**	**0.9264**	🥇 1st
XGBoost	0.2549	0.9259	🥈 2nd
Random Forest	0.2828	0.9178	🥉 3rd
SVR	0.3451	0.8997	4th
Linear Regression	0.7589	0.7794	5th

6.2 Performance Interpretation

Winning Model: LightGBM

- **MSE = 0.2533**: Average squared error of ~0.25 days²
- **R^2 = 0.9264**: Explains 92.64% of variance in shipping times
- **Strengths**: Fast training, excellent accuracy, handles non-linear relationships

Runner-up: XGBoost

- **MSE = 0.2549**: Very close to LightGBM performance
- **R^2 = 0.9259**: Explains 92.59% of variance
- **Strengths**: Robust, well-regularized, industry standard

Baseline: Linear Regression

- **MSE = 0.7589**: Highest prediction error
- **R^2 = 0.7794**: Explains only 77.94% of variance
- **Limitation**: Cannot capture non-linear feature interactions

6.3 Key Insights

1. **Non-linear Relationships**: Tree-based models significantly outperform linear regression
2. **Feature Interactions**: Gradient boosting captures complex feature combinations
3. **Minimal Overfitting**: Close performance between LightGBM and XGBoost suggests good generalization
4. **Feature Engineering Success**: One-hot encoding effectively handled categorical variables

---

7. Model Predictions and Validation

7.1 Sample Prediction Example

Input Features:

```
Distance: 1016 km
Products Sold: 254
Transportation: Air (encoded as 0,0,0 for Rail,Road,Sea)
Location: Mumbai (encoded as 0,0,0,1 for Chennai,Delhi,Kolkata,Mumbai)
Route: Route B (encoded as 1,0 for Route B, Route C)
```

Predictions:

- **Linear Regression**: 3.00 days
- **Random Forest**: 2.85 days
- **SVR**: 2.09 days
- **LightGBM**: 2.51 days
- **XGBoost**: 2.50 days

7.2 Individual Row Analysis

Testing on specific data points (rows 0, 2, 4):

Row 0 Performance:

- Actual: 3 days
- Linear Regression: 3 days (✓ Perfect)
- Random Forest: 3 days (✓ Perfect)
- SVR: 2 days (✗ -1 day error)
- LightGBM: 3 days (✓ Perfect)
- XGBoost: 2 days (✗ -1 day error)

Key Observation: Individual predictions may vary, but overall metrics show gradient boosting models are superior across the full dataset.

---

8. Technical Implementation Details

8.1 Code Structure

```
Project Structure:
├── Data Loading (shipment_df.csv)
├── Feature Engineering (One-hot encoding)
├── Data Preprocessing (Scaling, Train-test split)
├── Model Training (5 algorithms)
├── Performance Evaluation (MSE, R²)
└── Prediction Testing (Sample data)
```

8.2 Key Libraries and Versions

```python
import pandas as pd                    # Data manipulation
import matplotlib.pyplot as plt        # Visualization
import seaborn as sns                  # Enhanced plotting
import numpy as np                     # Numerical operations
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
```

8.3 Reproducibility

- **Random States**: Set to 30 (train_test_split) and 32 (models)
- **Fixed Parameters**: All hyperparameters explicitly defined
- **Environment**: Python 3.x with standard data science libraries

---

9. Business Impact and Applications

9.1 Practical Applications

1. **Customer Service**: Accurate delivery time estimates
2. **Logistics Planning**: Resource allocation optimization
3. **Inventory Management**: Better stock level predictions
4. **Cost Optimization**: Route and mode selection guidance

9.2 Economic Value

- **Reduced Customer Complaints**: More accurate delivery promises
- **Operational Efficiency**: Better resource planning
- **Competitive Advantage**: Superior service reliability
- **Cost Savings**: Optimized transportation decisions

9.3 Scalability Considerations

- **Real-time Predictions**: Models can be deployed for live predictions
- **Feature Expansion**: Additional variables (weather, traffic) can be incorporated
- **Model Updates**: Periodic retraining with new data
- **Integration**: API deployment for business system integration

---

10. Limitations and Future Work

10.1 Current Limitations

1. **Feature Set**: Limited to 6 basic features
2. **Temporal Aspects**: No time-series or seasonal considerations
3. **External Factors**: Weather, traffic, holidays not included
4. **Model Complexity**: Basic hyperparameter settings used

10.2 Recommended Improvements

1. **Feature Engineering**:

- Add temporal features (day of week, month, season)
- Include external data (weather, traffic conditions)
- Create interaction features between existing variables

2. **Model Enhancement**:
   - Hyperparameter tuning using GridSearch or Bayesian optimization
   - Cross-validation for more robust performance estimates
   - Ensemble methods combining multiple models
3. **Advanced Techniques**:
   - Neural network implementation
   - Time series forecasting models
   - Deep learning approaches for complex pattern recognition
4. **Validation Strategy**:
   - K-fold cross-validation
   - Time-based validation splits
   - Out-of-sample testing on different time periods

---

11. Conclusions

11.1 Key Achievements

✅ Successfully implemented 5 different machine learning models
✅ Achieved 92.64% accuracy in shipping time prediction
✅ Properly handled categorical and numerical features
✅ Demonstrated clear performance comparisons
✅ Created production-ready prediction pipeline

11.2 Main Findings

1. **Gradient Boosting Superior**: LightGBM and XGBoost significantly outperformed other methods
2. **Non-linear Relationships**: Complex feature interactions exist in shipping data
3. **Feature Engineering Critical**: Proper encoding and scaling essential for performance
4. **Model Selection Matters**: 15+ percentage point difference between best and worst models

11.3 Recommendations for Production

1. **Deploy LightGBM Model**: Best balance of accuracy and speed

2. **Implement Real-time Pipeline**: For live shipping time predictions
3. **Monitor Model Performance**: Set up automated retraining triggers
4. **Expand Feature Set**: Incorporate additional relevant variables
5. **A/B Testing**: Compare model predictions against current estimation methods

---

12. Appendices

Appendix A: Complete Performance Metrics

Model	MSE	R^2 Score
Linear Regression	0.7589	0.7794
Random Forest	0.2828	0.9178
SVR	0.3451	0.8997
LightGBM	0.2533	0.9264
XGBoost	0.2549	0.9259

Appendix B: Feature Encoding Mapping

transportation_modes:
- Air → (0,0,0) for Rail,Road,Sea
- Rail → (1,0,0) for Rail,Road,Sea
- Road → (0,1,0) for Rail,Road,Sea
- Sea → (0,0,1) for Rail,Road,Sea

location:
- Mumbai → (0,0,0,1) for Chennai,Delhi,Kolkata,Mumbai
- Chennai → (1,0,0,0) for Chennai,Delhi,Kolkata,Mumbai
- Delhi → (0,1,0,0) for Chennai,Delhi,Kolkata,Mumbai
- Kolkata → (0,0,1,0) for Chennai,Delhi,Kolkata,Mumbai

routes:
- Route A → (0,0) for Route B, Route C
- Route B → (1,0) for Route B, Route C
- Route C → (0,1) for Route B, Route C

Appendix C: Model Hyperparameters Summary

python

```
models_config = {
    "Linear Regression": {"default": "no parameters"},
    "Random Forest": {"random_state": 32, "default_trees": 100},
    "SVR": {"kernel": "rbf", "C": 10, "epsilon": 0.1},
    "LightGBM": {"n_estimators": 5, "learning_rate": 0.6,
                 "max_depth": 7, "random_state": 32},
    "XGBoost": {"n_estimators": 100, "learning_rate": 0.1,
                "max_depth": 5, "random_state": 32}
}
```

Project Completed By: FlexSaf Intern

Date: 2024

Technology Stack: Python, Scikit-learn, LightGBM, XGBoost

Final Model Recommendation: LightGBM (MSE: 0.2533, R^2: 0.9264)