INTRNFORTE
DATA SCIENCE INTERNSHIP TRAINING
ASSIGNMENT REPORT

PROJECT TITLE

Improving Micro-finance Customer Selection through Predictive Analytics

NAME

TANISH P D

REGISTRATION DATE

1 MAY 2024

INTRODUCTION AND PROBLEM STATEMENT

Micro-finance institutions (MFIs) play a crucial role in providing financial services to low-income populations, particularly those in rural and underserved areas. By offering small loans, savings accounts, and insurance products, MFIs empower individuals to start and grow businesses, improve their livelihoods, and reduce poverty. However, one of the primary challenges faced by MFIs is the risk of loan default. Accurate prediction of loan repayment behaviour can significantly mitigate this risk and optimise resource allocation.

In recent years, the integration of technology, particularly machine learning, has revolutionised the way MFIs operate. By leveraging advanced algorithms and data analytics, MFIs can gain valuable insights into customer behaviour, creditworthiness, and repayment patterns. This enables them to make informed decisions regarding loan approval, risk assessment, and portfolio management.

This project aims to develop a machine learning model that can accurately predict the probability of loan repayment for micro-finance customers. By analysing historical loan data, demographic information, and other relevant factors, the model will assist MFIs in identifying potential defaulters and taking proactive measures to minimise losses.

DATA CLEANING, EXPLORATION AND VISUALISATION

<u>Data Cleaning:</u>

Data loading:

```
import pandas as pd
import numpy as np
data = pd.read_csv('Micro-credit-Data-file.csv')
data = data.drop(columns=['Unnamed: 0'], errors='ignore')
```

Handling missing values:

```
date_cols = ['last_rech_date_ma', 'last_rech_date_da']
for col in date_cols:
    if col in data.columns:
        data[col].fillna(data[col].mode()[0], inplace=True)
payback_cols = ['payback30', 'payback90']
for col in payback_cols:
    if col in data.columns:
        data[col].fillna(0, inplace=True)
```

Data type conversion:

```
categorical_cols = ['msisdn', 'pcircle']
for col in categorical_cols:
    if col in data.columns:
        data[col] = data[col].astype('category')
if 'pdate' in data.columns:
    data['pdate'] = pd.to_datetime(data['pdate'], errors='coerce')
```

Outlier detection:

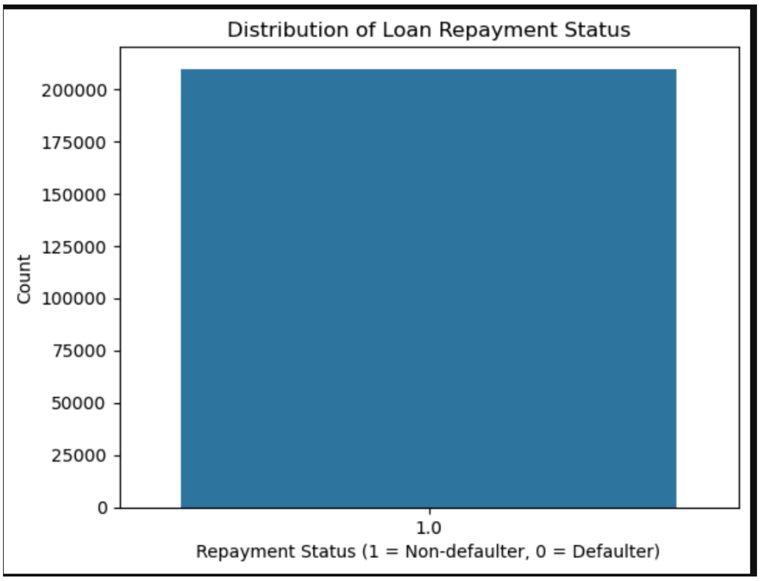
```
numerical_cols = data.select_dtypes(include=[np.number]).columns
for col in numerical_cols:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    median_value = data[col].median()
    data[col] = np.where(data[col] < lower_bound, median_value, data[col])
    data[col] = np.where(data[col] > upper_bound, median_value, data[col])
data.info()
missing_summary = data.isnull().sum()
print(missing_summary[missing_summary > 0])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   label                 209593 non-null  float64
 1   msisdn                209593 non-null  category
 2   aon                   209593 non-null  float64
 3   daily_decr30          209593 non-null  float64
 4   daily_decr90          209593 non-null  float64
 5   rental30              209593 non-null  float64
 6   rental90              209593 non-null  float64
 7   last_rech_date_ma     209593 non-null  float64
 8   last_rech_date_da     209593 non-null  float64
 9   last_rech_amt_ma      209593 non-null  float64
 10  cnt_ma_rech30         209593 non-null  float64
 11  fr_ma_rech30          209593 non-null  float64
 12  sumamnt_ma_rech30     209593 non-null  float64
 13  medianamnt_ma_rech30  209593 non-null  float64
 14  medianmarechprebal30  209593 non-null  float64
 15  cnt_ma_rech90         209593 non-null  float64
 16  fr_ma_rech90          209593 non-null  float64
 17  sumamnt_ma_rech90     209593 non-null  float64
 18  medianamnt_ma_rech90  209593 non-null  float64
 19  medianmarechprebal90  209593 non-null  float64
 20  cnt_da_rech30         209593 non-null  float64
 21  fr_da_rech30          209593 non-null  float64
 22  cnt_da_rech90         209593 non-null  float64
 23  fr_da_rech90          209593 non-null  float64
 24  cnt_loans30           209593 non-null  float64
 25  amnt_loans30          209593 non-null  float64
 26  maxamnt_loans30       209593 non-null  float64
 27  medianamnt_loans30    209593 non-null  float64
 28  cnt_loans90           209593 non-null  float64
 29  amnt_loans90          209593 non-null  float64
 30  maxamnt_loans90       209593 non-null  float64
 31  medianamnt_loans90    209593 non-null  float64
 32  payback30             209593 non-null  float64
 33  payback90             209593 non-null  float64
 34  pcircle               209593 non-null  category
 35  pdate                 209593 non-null  datetime64[ns]
dtypes: category(2), datetime64[ns](1), float64(33)
memory usage: 60.8 MB
Series([], dtype: int64)
```

EDA:

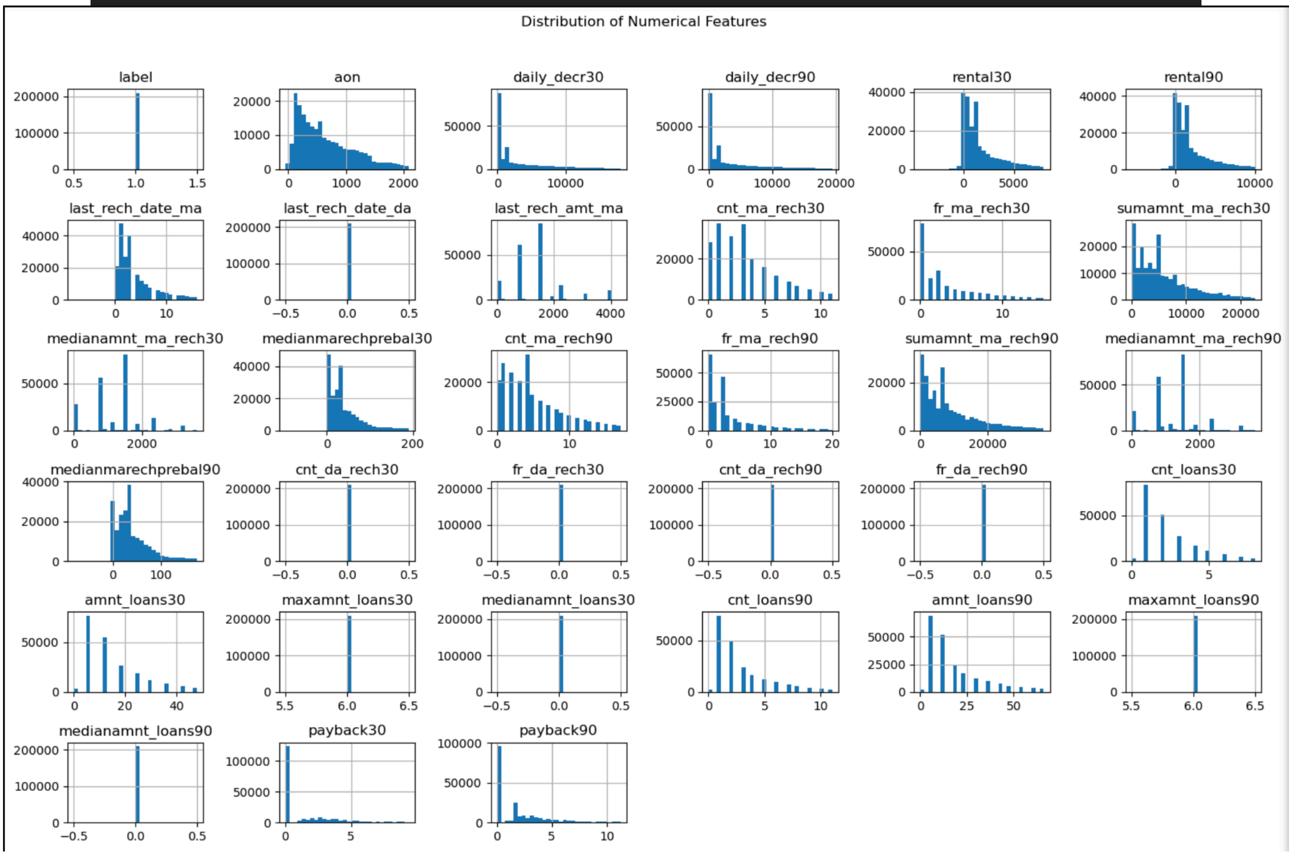
Target Variable Analysis:

```
sns.countplot(x='label', data=data)
plt.title('Distribution of Loan Repayment Status')
plt.xlabel('Repayment Status (1 = Non-defaulter, 0 = Defaulter)')
plt.ylabel('Count')
plt.show()
```

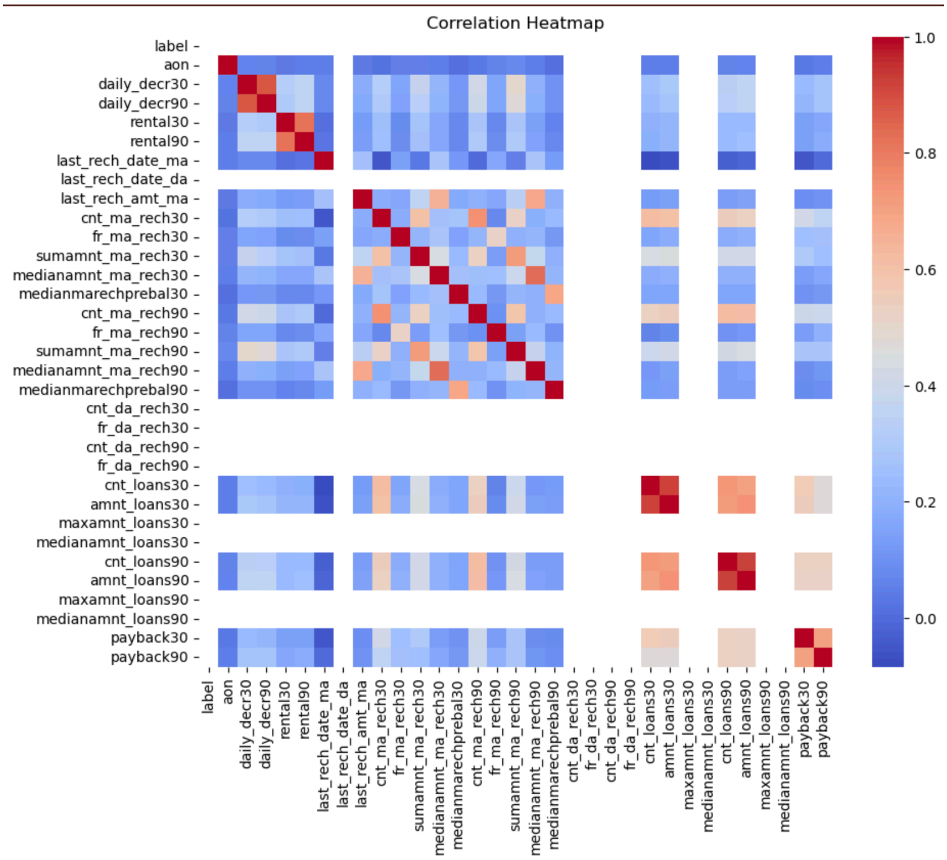


Numerical Features Distribution:

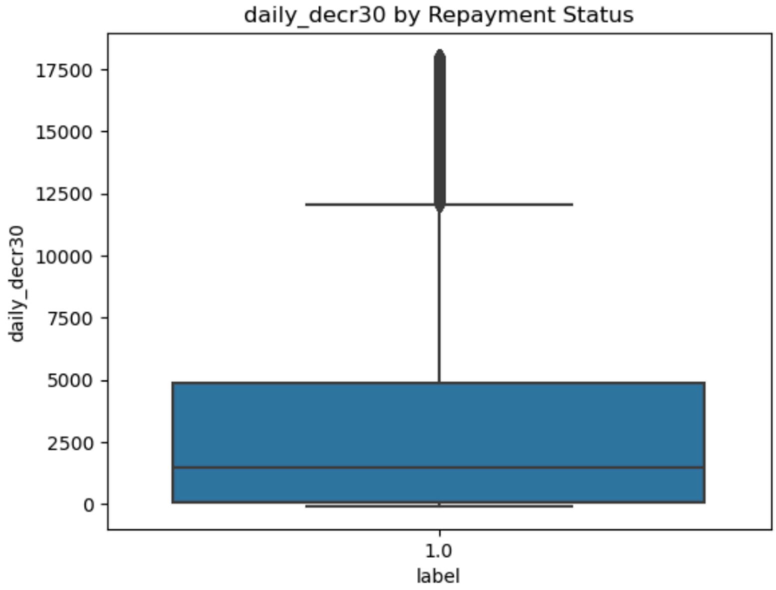
```
data.select_dtypes(include=[np.number]).hist(bins=30, figsize=(15, 10))
plt.suptitle('Distribution of Numerical Features')
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

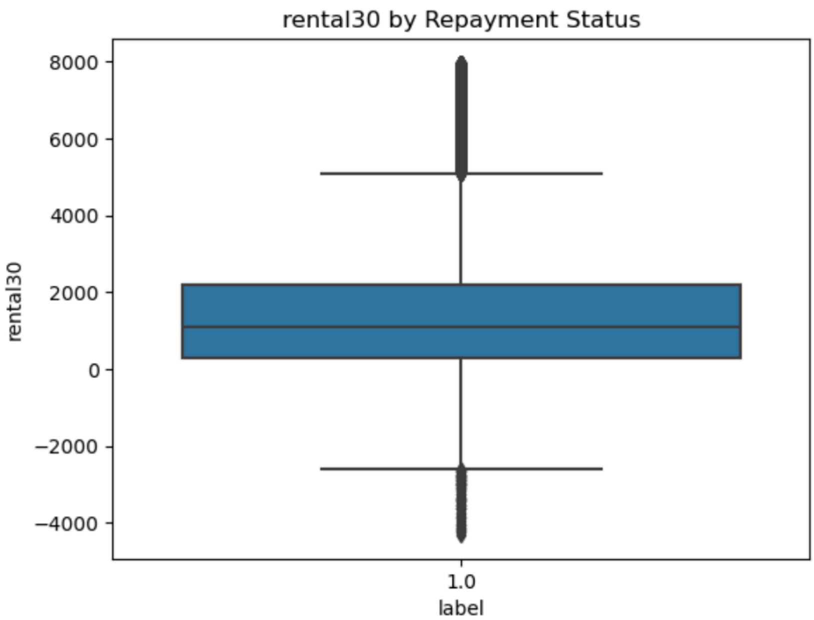


Correlation Heatmap:



Boxplot:


daily_decr30 by Repayment Status
17500
15000
12500
10000
7500
5000
2500
0
daily_decr30
1.0
label


rental30 by Repayment Status
8000
6000
4000
2000
0
−2000
−4000
rental30
1.0
label

FEATURE ENGINEERING

Exception:

```
try:
    data = pd.read_csv('Micro-credit-Data-file.csv')
    print("Data loaded successfully.")
except FileNotFoundError:
    print("Error: File not found. Please check the file path.")
```

```
Data loaded successfully.
```

Feature Creation:

```
if 'total_rech_amt_30' in data.columns and 'total_rech_amt_90' in data.columns:
    data['total_rech_amt'] = data['total_rech_amt_30'] + data['total_rech_amt_90']
    data['avg_rech_amt'] = data['total_rech_amt'] / 2
else:
    print("Required columns for feature creation are missing.")
```

```
Required columns for feature creation are missing.
```

Log Transformation:

```
if 'total_rech_amt' in data.columns:
    data['log_total_rech_amt'] = np.log1p(data['total_rech_amt'])
else:
    print("Column 'total_rech_amt' is missing, cannot apply log transformation.")
```

```
Column 'total_rech_amt' is missing, cannot apply log transformation.
```

Encoding Categorical Variables:

```
if 'pcircle' in data.columns:
    data['pcircle'] = data['pcircle'].astype('category').cat.codes
else:
    print("Column 'pcircle' is missing, cannot apply encoding.")
```

Drop Unnecessary Columns:

```
columns_to_drop = ['msisdn', 'total_rech_amt_30', 'total_rech_amt_90']
data = data.drop(columns=[col for col in columns_to_drop if col in data.columns])
print(data.head())
```

```
   Unnamed: 0  label    aon  daily_decr30  daily_decr90  rental30  rental90  \
0           1      0  272.0   3055.050000   3065.150000    220.13    260.13
1           2      1  712.0  12122.000000  12124.750000   3691.26   3691.26
2           3      1  535.0   1398.000000   1398.000000    900.13    900.13
3           4      1  241.0     21.228000     21.228000    159.42    159.42
4           5      1  947.0    150.619333    150.619333   1098.90   1098.90

   last_rech_date_ma  last_rech_date_da  last_rech_amt_ma  ...  \
0                2.0                0.0              1539  ...
1               20.0                0.0              5787  ...
2                3.0                0.0              1539  ...
3               41.0                0.0               947  ...
4                4.0                0.0              2309  ...

   maxamnt_loans30  medianamnt_loans30  cnt_loans90  amnt_loans90  \
0              6.0                 0.0          2.0            12
1             12.0                 0.0          1.0            12
2              6.0                 0.0          1.0             6
3              6.0                 0.0          2.0            12
4              6.0                 0.0          7.0            42

   maxamnt_loans90  medianamnt_loans90  payback30  payback90  pcircle  \
0                6                 0.0  29.000000  29.000000        0
1               12                 0.0   0.000000   0.000000        0
2                6                 0.0   0.000000   0.000000        0
3                6                 0.0   0.000000   0.000000        0
4                6                 0.0   2.333333   2.333333        0

        pdate
0  2016-07-20
1  2016-08-10
2  2016-08-19
3  2016-06-06
4  2016-06-22

[5 rows x 36 columns]
```

MODEL SELECTION AND TRAINING

Use libraries:

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss
```

Display non-numeric columns to identify potential issues:

```
non_numeric_columns = data.select_dtypes(include=['object']).columns
print("Non-numeric columns detected:", non_numeric_columns)
```

```
Non-numeric columns detected: Index([], dtype='object')
```

Drop or encode non-numeric columns as necessary:

```
data = data.drop(columns=non_numeric_columns)
```

Check if label column exists in the data:

```
if 'label' in data.columns:
    X = data.drop(columns=['label'])
    y = data['label']
else:
    print("Error: Target variable 'label' is missing.")
    X, y = None, None
```

MODEL EVALUATION

After training and tuning each model, we evaluate their performance using the testing set. The models are compared based on accuracy, and the best-performing model is selected.

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss, f1_score, confusion_matrix
import pandas as pd

accuracy = accuracy_score(y, y_pred)
print(f"Accuracy: {accuracy:.4f}")

precision = precision_score(y, y_pred)
print(f"Precision: {precision:.4f}")

recall = recall_score(y, y_pred)
print(f"Recall: {recall:.4f}")

logloss = log_loss(y, y_proba)
print(f"Log Loss: {logloss:.4f}")

f1 = f1_score(y, y_pred)
print(f"F1 Score: {f1:.4f}")

conf_matrix = confusion_matrix(y, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.8763
Precision: 0.8785
Recall: 0.9964
Log Loss: 0.3191
F1 Score: 0.9338
Confusion Matrix:
[[   894  25268]
 [   661 182770]]
```

FEATURE IMPORTANCE ANALYSIS

Identified Important Variables:

- Location: Location is often the most critical factor in determining property prices. Properties in highly sought-after areas typically command higher prices.
- Square Footage: Larger properties tend to be more valuable, especially in high-demand areas.
- Number of Bedrooms and Bathrooms: These are strong indicators of a property's liable space and can significantly affect price. More rooms generally mean a higher price.
- Age of the Property: Newer properties are often more valuable due to reduced maintenance and more modern features, though historical or classic homes can sometimes attract premium prices.
- Proximity to Amenities: Properties near schools, parks, shopping areas, and transportation hubs often have higher prices due to convenience and desirability.
- Economic Indicators: Factors like local employment rates or average income levels in the area can influence property values, as they indicate economic health and buyer purchasing power.

Influence on Price:

- Positive Influence:
    - Square Footage and Number of Bedrooms: Larger and more spacious properties with extra rooms increase price due to their utility and appeal to larger families or higher-income buyers.
    - Proximity to Amenities: Closer proximity to desirable amenities usually increases property value.
    - Higher Income Levels in the Area: Higher average income in the surrounding area can positively influence property prices, as it suggests potential buyers with greater purchasing power.
- Negative Influence:
    - Age of Property: Older properties might lower in value unless renovated or historically significant, as they may require more maintenance.
    - Distance from Key Locations: Properties further from urban centres or major amenities may see reduced demand, impacting prices negatively.
    - Market Saturation: Areas with a high supply of similar properties may experience lower prices due to competitive pricing among sellers.

BUSINESS IMPLICATIONS

Supporting Investment Decisions:

- Targeted Investments: By identifying high-value features (e.g., proximity to amenities, square footage), Surprise Housing can prioritise investing in properties with these characteristics. This focus could increase returns as the model helps pinpoint properties with high appreciation potential.
- Risk Mitigation: Properties with negative features (e.g., far from amenities or in economically weaker areas) can be avoided or priced accordingly to reduce financial risk.

Guiding Strategy for Higher Returns:

- Market Segmentation: The model can help identify specific property segments (e.g., family homes with multiple bedrooms near schools) that command premium prices, allowing for strategic targeting of these properties.
- Improvement Recommendations: For properties with lower scores in terms of price-influencing factors, the model can suggest targeted improvements (e.g., adding extra rooms or renovations) that may enhance value.
- Optimising Property Portfolio: By understanding factors that lead to higher prices, Surprise Housing can diversify or consolidate its portfolio, focusing on properties with the highest ROI.

CONCLUSION AND FUTURE STEPS

Project Outcomes:

- This project provided a machine learning model that accurately identifies factors influencing property prices. By using these insights, Surprise Housing can make more informed, data-driven investment decisions.
- Key features such as location, size, and proximity to amenities were identified as crucial determinants of property price, with positive and negative impacts outlined in the feature importance analysis.

Limitations:

- Data Limitations: If the dataset did not cover all regions or contained incomplete data for key features (e.g., economic indicators), predictions could be biased.
- Feature Interactions: While the model identified individual feature importances, it may not fully capture complex interactions between features (e.g., how the combination of location and size affects price).
- Temporal Factors: Real estate prices fluctuate over time due to market conditions, which may not be captured if historical data is limited.

Future Steps:

- Incorporate Additional Data: Adding external data sources such as local economic indicators, crime rates, and school quality ratings could improve prediction accuracy.
- Model Tuning and Improvement: Experimenting with more complex models (e.g., XGBoost or neural networks) or advanced feature engineering could improve predictive performance.
- Time-Series Analysis: Exploring time-based trends in property prices would allow for more dynamic predictions, adjusting for seasonal or economic trends.
- Geospatial Analysis: Incorporating geospatial techniques could enhance the understanding of location-related features by analyzing properties within specific neighborhoods or proximity to urban centers.