MINISTRY OF EDUCATION AND TRAINING

EASTERN INTERNATIONAL UNIVERSITY

MIS 395

Artificial Intelligence for Business

Final Project

Business Applications of Predictive Analytics with AI-Based Tools

Lecturers: ***Mr. Dang Thai Doan***

GROUP: 3A	
Name	**IRN**
Nguyễn Thành Đạt	2132300562
Bùi Thị Thanh Thảo	2232300157
Lê Nguyễn Tâm Như	2132300065

Quarter 4/2024-2025

Date Submission: 13/8/2025

Table of Contents

I. Dataset: Superstore Sales Data

1. Dataset overview

The dataset used in this analysis belongs to the marketing domain and was obtained from a publicly available source. It contains 2,237 records and 26 variables, which include customer demographic information, purchasing behavior across various product categories, and responses to marketing campaigns. The main goal of this analysis is to predict total customer spending and identify some factors that influence spending behavior. The dataset was preprocessed using Google Colab for data cleaning and RapidMiner for modeling and visualization.

2. Data preprocessing

The data preprocessing phase was conducted primarily in Google Colab using Python. The first step involved converting the Dt_Customer column into a standard datetime format to facilitate future time-based analysis. Then, a new column Age was computed by subtracting the year of birth from the current year (2025). To analyze overall purchasing behavior, a new feature called Total_Spending was created by summing customer spending across six product categories, including wines, fruits, meat products, fish, sweets, and gold products.

Next, a normalization step was applied using the MinMaxScaler from the sklearn.preprocessing module. This transformation produced two new variables: Income_Norm and Total_Spending_Norm, which rescale the income and total spending values into a [0, 1] range. This was important to ensure fair weighting of attributes in later modeling processes.

Data filtering was then performed to remove extreme or unrealistic values. Specifically, records with a Year_Birth earlier than 1920 were excluded to eliminate overly old individuals, and customers with an Age greater than or equal to 100 were also removed to focus on a more plausible target segment. Additionally, in the Marital_Status column, non-standard labels such as "YOLO" and "Absurd" were grouped under a new category called "Other" to simplify the classification task and avoid sparse categories.

```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], errors='coerce')

df['Age'] = 2025 - df['Year_Birth']

spending_cols = ['MntWines', 'MntFruits', 'MntMeatProducts',
                 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
df['Total_Spending'] = df[spending_cols].sum(axis=1)
```

```
[ ] from sklearn.preprocessing import MinMaxScaler

    scaler = MinMaxScaler()
    df[['Income_Norm', 'Total_Spending_Norm']] = scaler.fit_transform(df[['Income', 'Total_Spending']])
```

```
df = df[df['Year_Birth'] >= 1920]

df['Marital_Status'] = df['Marital_Status'].replace(['YOLO', 'Absurd'], 'Other')

df = df[df['Age'] < 100]
```

Finally, the cleaned and transformed dataset was saved locally as a CSV file named superstore_cleaned.csv and prepared for further analysis using RapidMiner. This step completed the essential preprocessing pipeline, ensuring a clean, well-structured, and machine-learning-ready dataset.

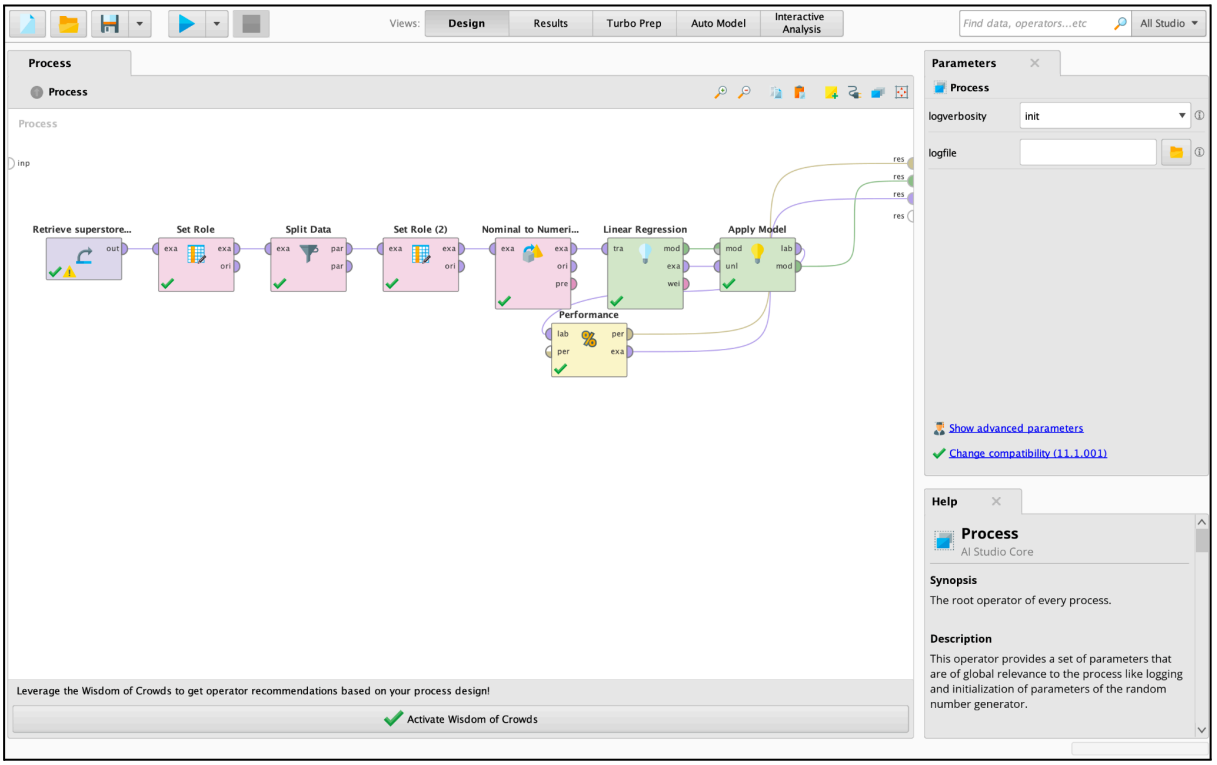
3. Exploratory Data Analysis (EDA)

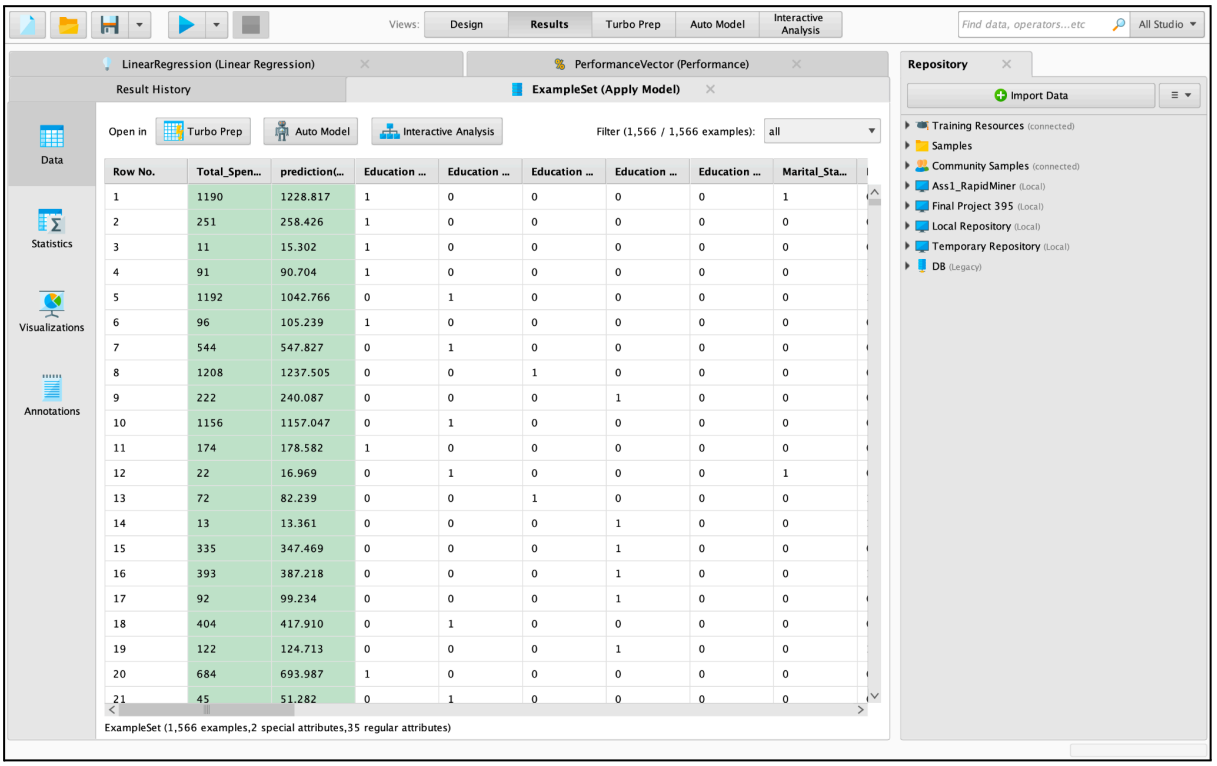
Exploratory data analysis revealed several key insights. The distribution of total spending was positively skewed, indicating a high frequency of lower spenders and a few high-value customers. Customers with higher income generally spent more across all categories. A breakdown by marital status showed that married and cohabiting individuals tend to spend more, particularly on wine and meat products. Additionally, higher educational levels appeared to correlate with an increase in catalog-based purchases. The EDA also included visualizations such as correlation matrices, histograms, and boxplots to further examine these relationships.

4. Descriptive Analytics

Five descriptive analyses were conducted. The first explored the relationship between age groups and total spending, with middle-aged customers (40–60 years old) emerging as the most valuable segment. The second analysis compared total spending across different marital statuses, confirming that married and "together" individuals had higher expenditures. Third, the impact of campaign response was evaluated, showing that customers who responded to marketing efforts generally spent more. Fourth, the breakdown of spending by

product category highlighted wine as the most purchased item. Finally, correlation analysis revealed a moderate positive relationship between income and total spending.

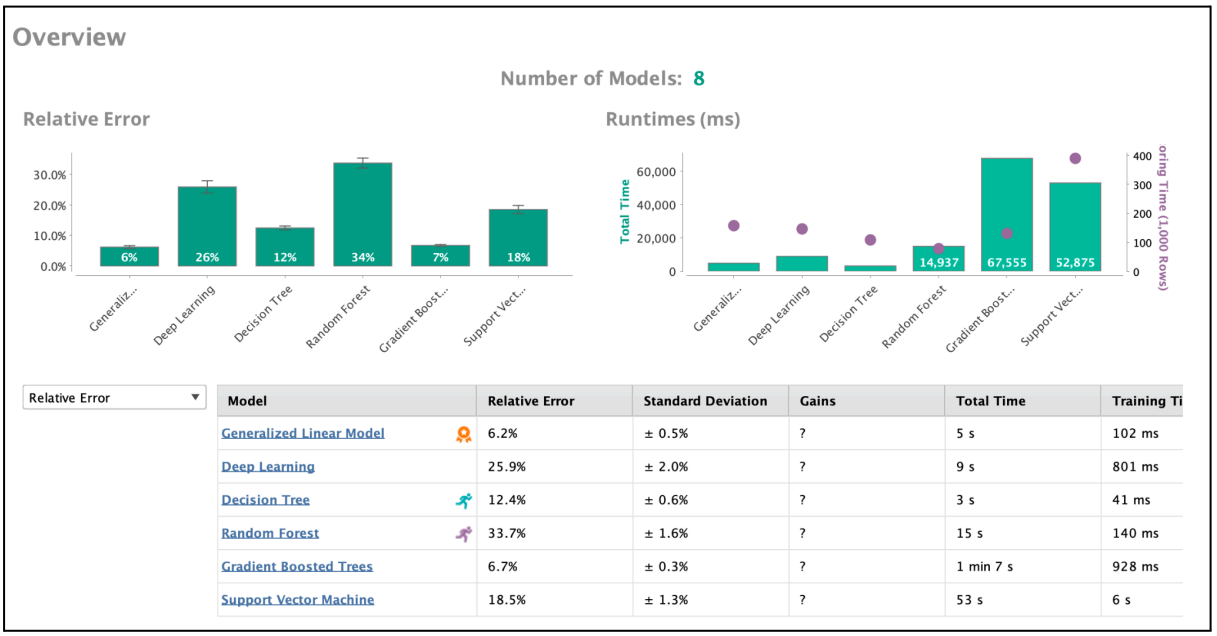




5. Tool Usage and Predictive Modeling

Two main tools were used in this analysis: Google Colab and RapidMiner. Google Colab was employed for initial preprocessing, feature engineering, and data transformation. RapidMiner was used for its powerful auto-modeling capabilities, allowing for efficient comparison of multiple regression models through its visual interface. Its built-in Auto Model feature provided quick insights into model performance, while allowing manual customization of processes. The use of these tools enabled a comprehensive, end-to-end workflow from raw data to actionable insights.

Multiple regression models were trained and evaluated in RapidMiner. These included Generalized Linear Model (GLM), Gradient Boosted Trees, Decision Tree, Support Vector Machine (SVM), Deep Learning, and Random Forest. Among these models, the Generalized Linear Model outperformed the others, achieving the lowest relative error of 6.2% and the fastest training time of just 5 seconds. In contrast, the Deep Learning model produced a relative error of 25.9%, while Random Forest had the highest error at 33.7%. The prediction chart for the GLM model showed a near-perfect alignment between actual and predicted values, indicating excellent model fit.



Relative Error

Model	Relative Error	Standard Deviation	Gains	Total Time	Training Ti
Generalized Linear Model	6.2%	± 0.5%	?	5 s	102 ms
Deep Learning	25.9%	± 2.0%	?	9 s	801 ms
Decision Tree	12.4%	± 0.6%	?	3 s	41 ms
Random Forest	33.7%	± 1.6%	?	15 s	140 ms
Gradient Boosted Trees	6.7%	± 0.3%	?	1 min 7 s	928 ms
Support Vector Machine	18.5%	± 1.3%	?	53 s	6 s

6. Business Insight from Dataset

The results provide practical insights for marketing strategy. Customers aged between 40 and 60 with moderate to high incomes are prime targets for future marketing campaigns.

Product categories such as wine and meat appear to be high-revenue generators and should be prioritized in promotional efforts. Furthermore, customers who responded positively to past campaigns were also among the highest spenders, suggesting that response history could be used for predictive targeting. The best-performing model (GLM) can be integrated into customer relationship management (CRM) systems to forecast spending behavior, personalize campaigns, and optimize customer segmentation. Despite its strengths, the dataset lacks temporal variables and geographic identifiers, which limits the depth of behavioral analysis. Future work could involve integrating time-series transactions or customer feedback for a more robust model.

Link to GitHub: https://github.com/ThaoBui195/MIS-451

II. Dataset: California Housing Prices

1. Dataset Overview

The California Housing Prices dataset is sourced from *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by Aurélien Géron, and it is widely available on platforms such as Kaggle and UCI. This dataset is frequently used in introductory machine learning courses for its simplicity and relevance in demonstrating key algorithms.

This dataset falls under the Real Estate domain and provides detailed information about housing prices in California. It includes features such as the median house value, the number of bedrooms, the total number of rooms, and geographical information. These attributes allow us to predict housing prices, identify market trends, and support decision-making processes in the real estate sector. By analyzing this data, one can gain valuable insights into the factors that affect housing prices, such as location and household characteristics.

With 20,640 rows and 10 columns, the dataset contains key features such as longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income, and median_house_value. One notable column, ocean_proximity, indicates the proximity of a house to the ocean, which is an important factor in determining the price of a property. This feature provides valuable context for analyzing how the proximity to the ocean influences property values in California.

2. Data Preprocessing & Management

2.1. Handling Missing Values

Upon inspecting the dataset, it was observed that one column, total_bedrooms, contains missing values (207 missing entries). These missing values can be addressed by using the mean imputation method. Specifically, we can impute the missing values with the mean of total_bedrooms grouped by the ocean_proximity feature. This ensures that the imputation respects the underlying distribution of the data based on the proximity to the ocean, which might influence the total number of bedrooms in a house. This approach avoids introducing any bias that might occur if we simply filled missing values with the overall mean of the column, and instead accounts for local variations in the dataset based on the proximity to the ocean.

2.2. Normalization and Scaling

Some numerical features in the dataset, such as total_rooms, total_bedrooms, population, households, and median_income, have different scales. To ensure that all features are treated equally, normalization was applied to adjust the values and bring them onto a similar scale. This is especially important for models like linear regression, where features with larger values can have more influence on the model's predictions.

After normalization, all numerical values were scaled so that each feature has a mean of 0 and a standard deviation of 1. This scaling ensures that no single feature dominates the model and helps improve the model's performance, especially during training, by speeding up its convergence.

2.3 Data Splitting

To prepare the data for training and testing the models, the dataset was split into two parts: training and test sets. 80% of the data was used for training, while the remaining 20% was held back for testing. This split allows for training the model on one set of data and validating its performance on unseen data, ensuring that the model generalizes well to new data.

2.4 Data Transformation

For the feature ocean_proximity which indicates the location of the house relative to the ocean, a method called one-hot encoding was used. This means that each possible location was turned into its own separate column with a simple 1 or 0. This allows the model to understand and work with the data without making assumptions about the relationship between different locations. By removing the first category, we avoid having extra unnecessary information, keeping the model clean and efficient.

3. Exploratory Data Analysis (EDA)

3.1 Descriptive Statistics

3.1.1. Before Normalization

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.880892	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	419.267735	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	297.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	438.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	643.250000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

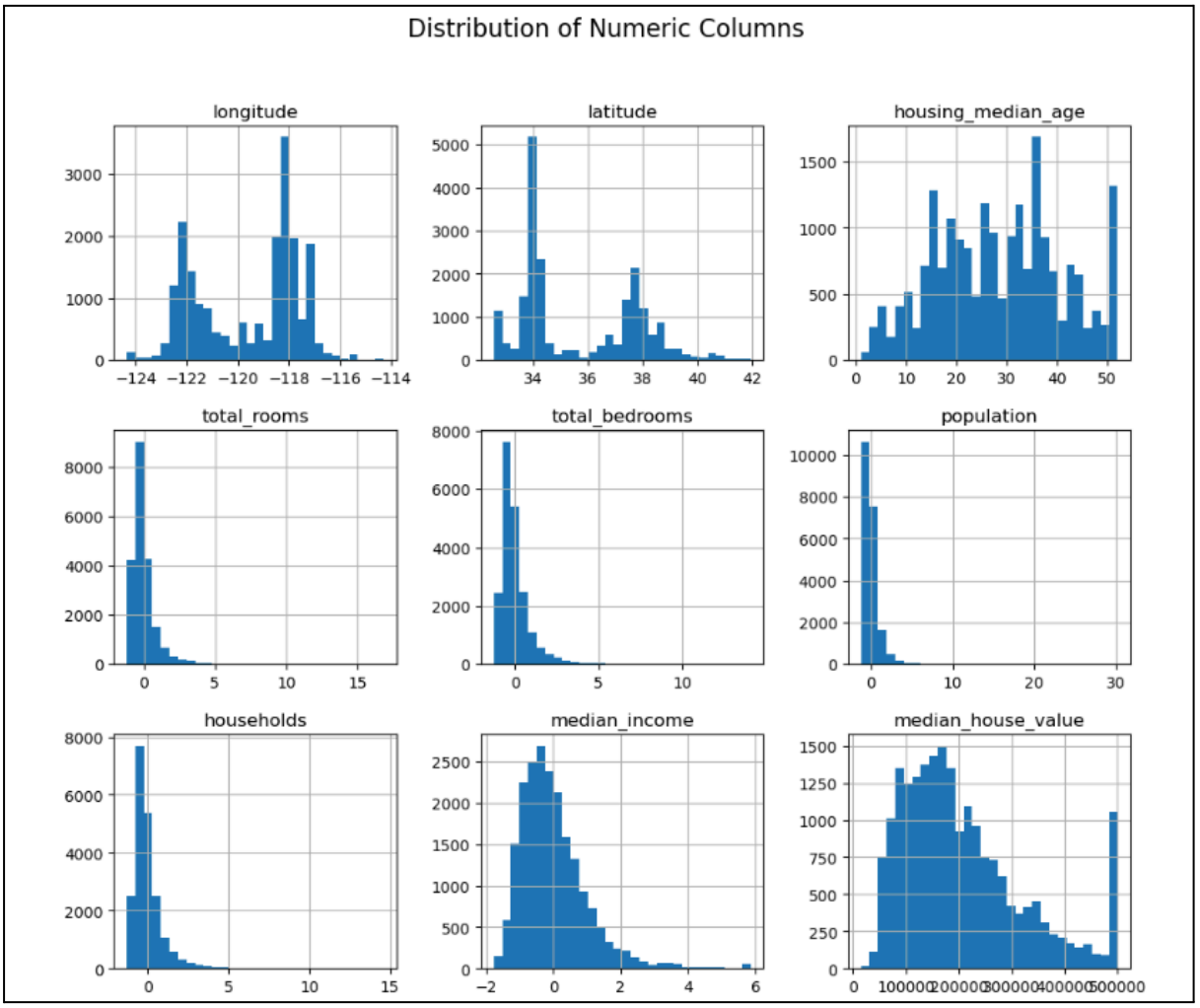
Descriptive statistics were calculated to summarize key features of the dataset, including count, mean, standard deviation, minimum, and maximum values. For example, "total_rooms" has a wide range from 2 to over 39,000, showing high variability, while "median_income" has a narrower range from 0 to 15, indicating more consistency. These statistics help identify the spread of the data and any potential outliers. Visualizations also aid in understanding feature distributions, highlighting where transformations may be needed before modeling.

3.1.2. After Normalization

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	20640.000000
mean	-119.569704	35.631861	28.639486	3.201573e-17	1.101617e-17	-1.101617e-17	6.885104e-17	6.609700e-17	206855.816909
std	2.003532	2.135952	12.585558	1.000024e+00	1.000024e+00	1.000024e+00	1.000024e+00	1.000024e+00	115395.615874
min	-124.350000	32.540000	1.000000	-1.207283e+00	-1.280551e+00	-1.256123e+00	-1.303984e+00	-1.774299e+00	14999.000000
25%	-121.800000	33.930000	18.000000	-5.445698e-01	-5.745415e-01	-5.638089e-01	-5.742294e-01	-6.881186e-01	119600.000000
50%	-118.490000	34.260000	29.000000	-2.332104e-01	-2.382328e-01	-2.291318e-01	-2.368162e-01	-1.767951e-01	179700.000000
75%	-118.010000	37.710000	37.000000	2.348028e-01	2.513231e-01	2.644949e-01	2.758427e-01	4.593063e-01	264725.000000
max	-114.310000	41.950000	52.000000	1.681558e+01	1.408947e+01	3.025033e+01	1.460152e+01	5.858286e+00	500001.000000

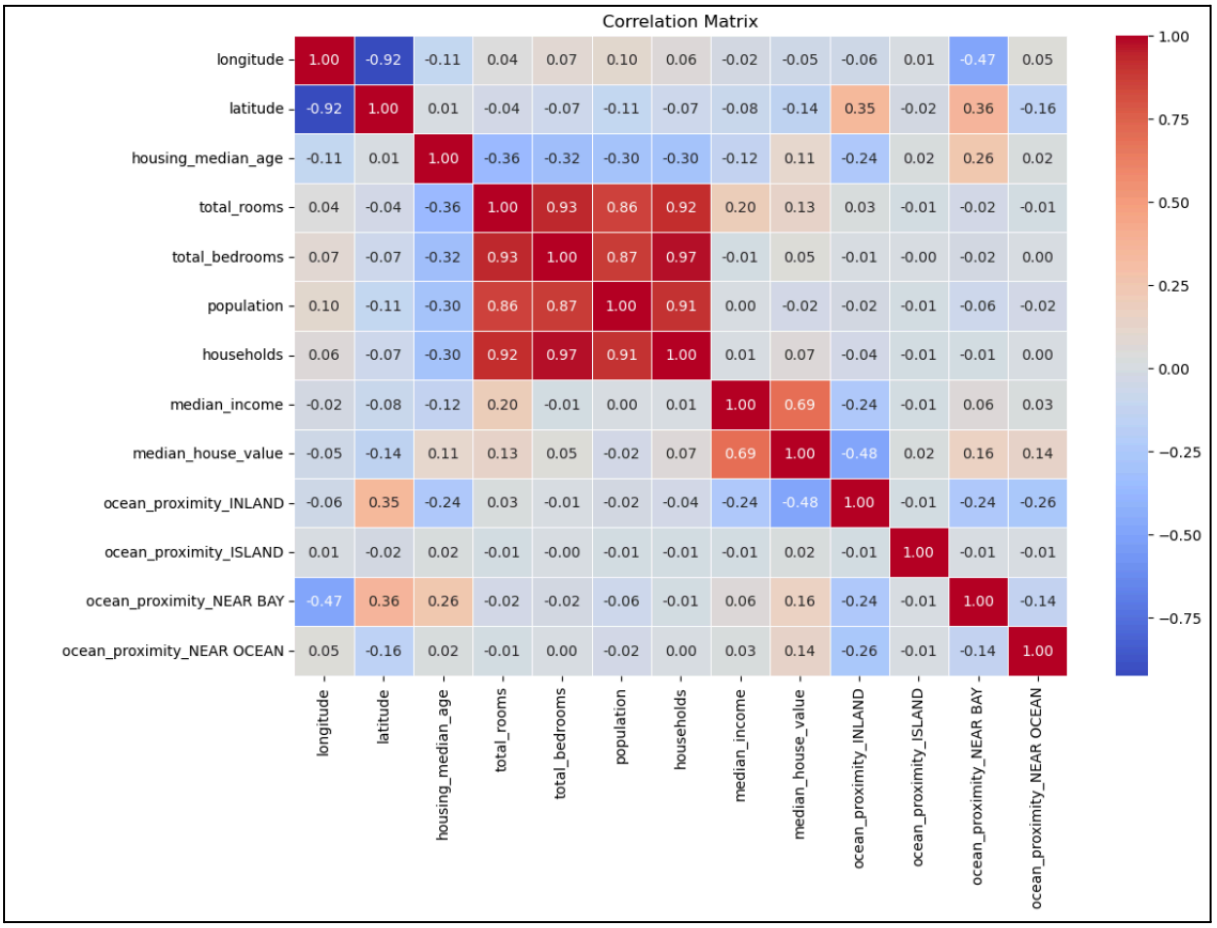
Normalization adjusted the features to have a mean of 0 and a standard deviation of 1, ensuring a standard distribution. This prevents any feature from dominating the model and speeds up training. After normalization, all features, including "total_rooms" and "total_bedrooms," were brought to a comparable scale. By connecting this with descriptive statistics, it's clear how normalization balances the features, enhancing the analysis and modeling process.

3.2. Feature Distributions



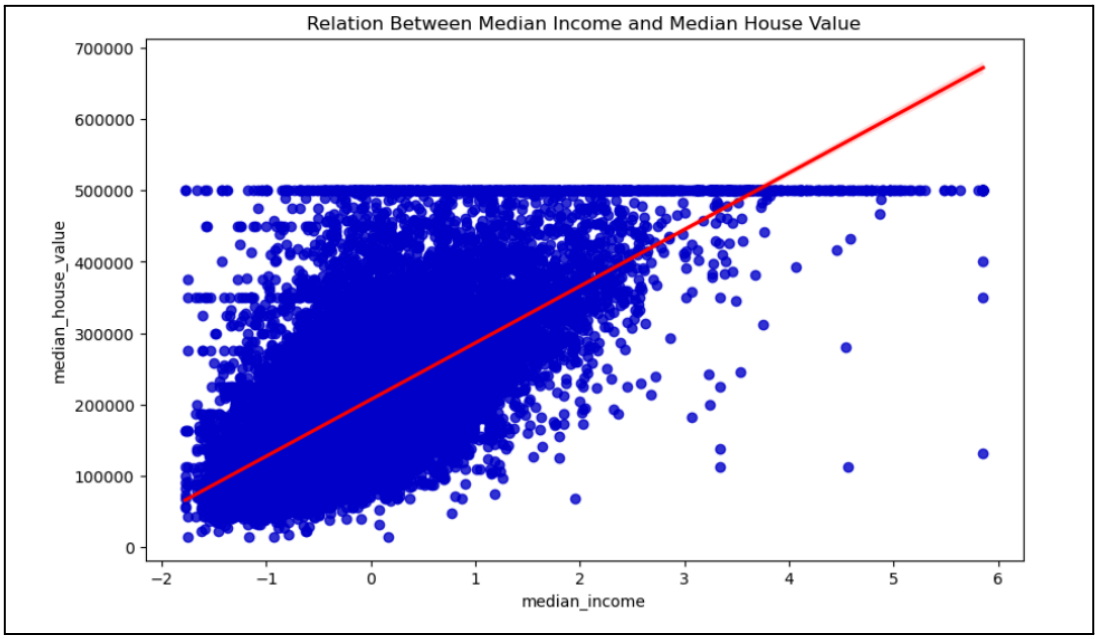
Feature Distributions were analyzed to understand how individual features behave. By using histograms, we observed that features such as total_rooms and total_bedrooms exhibit right-skewed distributions, indicating that most homes have a relatively small number of rooms and bedrooms, with fewer larger homes in the dataset. On the other hand, median_house_value showed a more normal distribution, which means that house prices in the dataset are more evenly spread out, though there are some outliers. These visualizations provided a deeper understanding of how each feature is distributed and whether any transformations would be needed for specific features before modeling.

3.3. Correlation Analysis



Correlation analysis revealed relationships between numerical features, providing insights into potential predictors. Total_rooms and total_bedrooms showed a strong positive correlation (0.93), as expected, since larger homes tend to have more bedrooms. Additionally, median_income was strongly correlated with median_house_value, indicating that higher-income areas generally have more expensive homes. Geographic features like longitude and latitude showed a negative correlation, reflecting California's geographical layout. The analysis also highlighted that homes near the ocean, especially those near the bay, tend to have more bedrooms, reflecting the premium on these properties.

3.4. Scatter Plot



Scatter plots were used to explore relationships between key features. The plot between median_income and median_house_value shows a clear upward trend, indicating that higher income is strongly linked to higher home prices. This suggests income is a key predictor for pricing.

Additionally, the plot between total_rooms and total_bedrooms shows a strong positive correlation, meaning homes with more rooms tend to have more bedrooms, which is a typical relationship in real estate.

4. Descriptive & Predictive Modeling

4.1. Model Selection

In this analysis, Linear Regression and Random Forest Regressor were chosen as predictive models. Linear Regression was selected for its simplicity and interpretability, making it ideal for predicting house prices based on features like median_income. Meanwhile, Random Forest was chosen for its ability to capture complex, non-linear relationships and handle large datasets with multiple interacting features. Both models were used without hyperparameter tuning or cross-validation, focusing on assessing basic predictive performance.

4.2 Performance Metrics

To evaluate the predictive performance of both models, three common metrics were used: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared. These metrics provide insights into how well the models fit the data.

4.2.1. For Random Forest:

- MAE: 31,670.37
- MSE: 24,036,753,19.9
- R-squared: 0.8166

The Random Forest model explains approximately 81.66% of the variance in the target variable (median_house_value). Despite relatively high MAE and MSE, the model performs well, particularly in capturing complex, non-linear relationships.

4.2.2. For Linear Regression:

- MAE = 50,704.38
- MSE = 49,049,024,02.99
- R-squared = 0.63

The Linear Regression model explains 62.57% of the variance in median_house_value. While its R-squared is lower than Random Forest, it provides a more

straightforward interpretation of variance. However, the higher MAE indicates larger prediction errors in some cases.

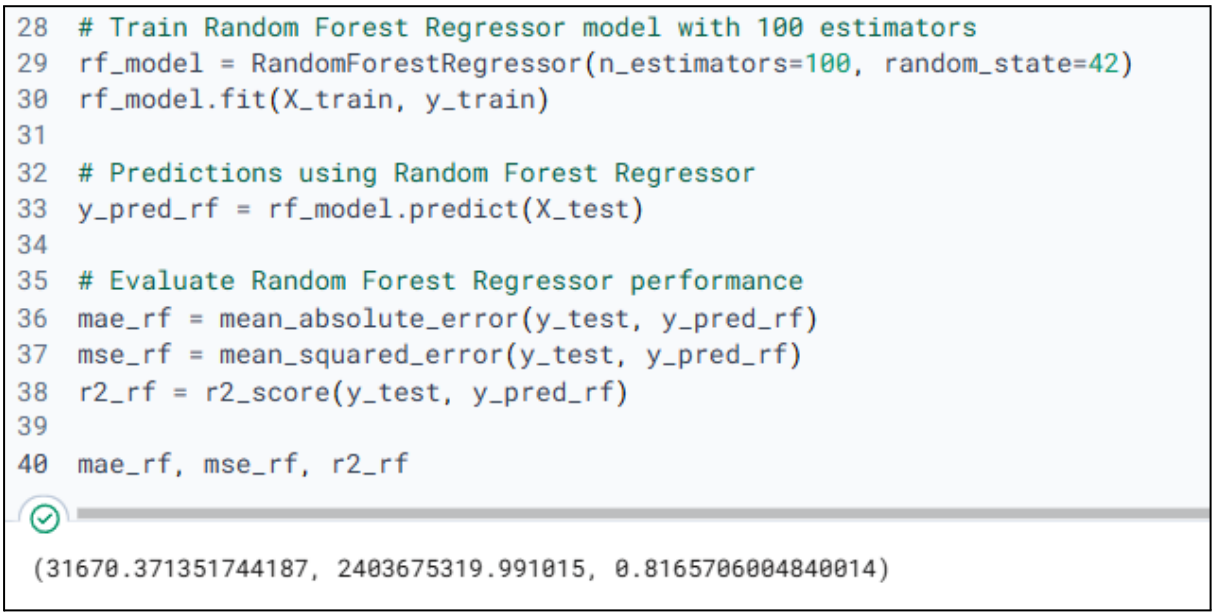
These performance metrics provide a comparative look at how each model performs. Although both models are reasonably accurate, Random Forest performs better in capturing complex patterns, whereas Linear Regression offers a clearer interpretation of the data.

5. **Tool Usage and Comparison**

DeepNote Link

In this dataset, both Deepnote and Jupyter were used to run the entire process, including data preprocessing, model training, and evaluation. Specifically, Deepnote utilized AI support to train both the Linear Regression and Random Forest models, similar to the process performed in Jupyter. Below are the results from both the Random Forest and Linear Regression models run on each tool.

For Deepnote:



For Jupyter Notebook:

```
#random forest
y_rf_pred = rf_model.predict(X_test)

rf_mae = mean_absolute_error(y_test, y_rf_pred)
rf_mse = mean_squared_error(y_test, y_rf_pred)
rf_r2 = r2_score(y_test, y_rf_pred)

print(f"Random Forest MAE: {rf_mae}")
print(f"Random Forest MSE: {rf_mse}")
print(f"Random Forest R-squared: {rf_r2}")
```

```
Random Forest MAE: 31670.371351744187
Random Forest MSE: 2403675319.991015
Random Forest R-squared: 0.8165706004840014
```

```
#linear regression
y_pred_all = model_all.predict(X_test_all)

mae_all = mean_absolute_error(y_test_all, y_pred_all)
mse_all = mean_squared_error(y_test_all, y_pred_all)
r2_all = r2_score(y_test_all, y_pred_all)

print(f"All Features - MAE: {mae_all}")
print(f"All Features - MSE: {mse_all}")
print(f"All Features - R-squared: {r2_all}")
```

```
All Features - MAE: 50704.38262926208
All Features - MSE: 4904902402.998749
All Features - R-squared: 0.6256968256137014
```

The results from both tools are identical, with metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared almost exactly the same. This shows that even though Deepnote has the advantage of AI support to speed up the process, the model results remain consistent across both tools. While Deepnote excels in speed and collaboration thanks to its cloud-based environment and AI, both tools demonstrate equivalent performance when it comes to model outcomes.

6. Insights for Business

The predictions generated by both models offer valuable insights for businesses in the real estate and property investment sectors. One key finding is the strong impact of median_income on median_house_value. The models show that higher household income is correlated with higher property values, indicating that areas with higher average incomes are more likely to have more expensive homes.

Understanding how features like total_rooms, total_bedrooms, and others correlate with housing prices can help businesses develop effective pricing strategies. For example, homes with more rooms or bedrooms generally have higher values, as predicted by the models. These insights can support decisions related to pricing, market segmentation, and help investors identify areas with growth potential based on housing characteristics.

Additionally, the results suggest that proximity to the ocean (represented by ocean_proximity) and the socio-economic context (including median_income) are significant predictors of home prices. This insight is especially valuable for property developers and real estate agents when identifying high-potential areas or creating targeted marketing strategies.

Link to Github

III. Diabetes 130 US hospitals for years 1999-2008

Graphite Note Link

1. Dataset Context and Objective

The dataset contains ten years (1999–2008) of clinical care records collected from 130 hospitals and integrated delivery networks across the United States. It comprises more than 50 characteristics that capture detailed information on patient demographics, diagnoses, treatments, and hospital-related outcomes.

The primary objective of this project is to develop a predictive model that determines the readmission status of diabetic patients. The target variable categorizes patients into three groups: those who are readmitted within 30 days, those readmitted after 30 days, and those not readmitted at all. This classification task holds significant value in healthcare settings, as

early identification of high-risk patients enables hospitals to implement timely interventions, reduce preventable readmissions, and allocate resources more effectively.

2. Dataset Overview

```
diabetes.shape
```

```
(101766, 50)
```

```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 50 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   encounter_id              101766 non-null  int64
 1   patient_nbr               101766 non-null  int64
 2   race                      101766 non-null  object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    101766 non-null  object
 6   admission_type_id         101766 non-null  int64
 7   discharge_disposition_id  101766 non-null  int64
 8   admission_source_id       101766 non-null  int64
 9   time_in_hospital          101766 non-null  int64
```

The dataset consists of 101,766 patient records and 50 attributes, offering a rich, structured view of inpatient encounters involving diabetic patients in the United States. These attributes cover a wide range of information, including demographic characteristics (such as race, gender, and age), hospitalization details (like admission type, discharge disposition, and number of inpatient visits), diagnostic codes, procedures, medications, and clinical outcomes.

3. Preprocessing

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	...	citoglipton	i
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	1	...	No	
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3	...	No	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2	...	No	
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2	...	No	
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1	...	No	S
5	35754	82637451	Caucasian	Male	[50-60)	?	2	1	2	3	...	No	S
6	55842	84259809	Caucasian	Male	[60-70)	?	3	1	2	4	...	No	S
7	63768	114882984	Caucasian	Male	[70-80)	?	1	1	7	5	...	No	
8	12522	48330783	Caucasian	Female	[80-90)	?	2	1	4	13	...	No	S
9	15738	63555939	Caucasian	Female	[90-100)	?	3	3	4	12	...	No	S

While the dataset originally appeared to have no missing values, further inspection reveals that 7 attributes contain missing data, encoded using the placeholder '?'. Therefore, I decided to replace '?' with null values.

```
missing_attributes = diabetes.isnull().sum()
missing_attributes = missing_attributes[missing_attributes > 0].sort_values(ascending=False)

print(missing_attributes)
```

```
weight                98569
max_glu_serum         96420
A1Cresult             84748
medical_specialty     49949
payer_code            40256
race                   2273
diag_3                 1423
diag_2                  358
diag_1                   21
dtype: int64
```

Specifically, the columns race, weight, payer_code, medical_specialty, and the three diagnosis fields (diag_1, diag_2, and diag_3) have missing values, with weight missing in over 97% of the entries, and others like medical_specialty and payer_code missing in roughly 40–50% of records.

```
diabetes.drop(columns=['encounter_id','patient_nbr','race',
                       'weight', 'payer_code', 'medical_specialty',
                       'max_glu_serum', 'A1Cresult'], inplace=True)
```

These columns were removed because they either contained a very high proportion of missing values or were irrelevant for predictive modeling. Specifically, *weight*, *payer_code*, *medical_specialty*, *max_glu_serum*, and *A1Cresult* had excessive missing data, which could introduce bias or reduce model reliability if imputed. *Encounter_id* and *patient_nbr* were unique identifiers with no predictive value, and *race* was excluded to avoid potential ethical concerns and bias in the model.

```python
def map_icd9(code):
    # Chuyển code sang string
    code = str(code)

    # Kiểm tra V-code
    if code.startswith('V'):
        return 'V-code'

    # Loại bỏ phần thập phân nếu có
    try:
        num = float(code)
    except:
        return 'Unknown'

    # Gom nhóm theo ICD-9 range
    if 1 <= num <= 139:
        return 'Infectious'
    elif 140 <= num <= 239:
        return 'Neoplasms'
```

The ICD-9 grouping code is a method used to categorize thousands of detailed diagnosis codes into broader, clinically meaningful disease groups. Instead of treating each individual ICD-9 code as a separate category, which can result in high cardinality and reduced model performance, this approach consolidates codes based on their standard ICD-9 classification ranges. For example, codes from 390 to 459 are grouped under circulatory diseases, while codes beginning with the letter "V" are assigned to a supplementary classification group representing factors influencing health status or reasons for healthcare encounters. Grouping in this way reduces complexity, improves model efficiency, and enhances interpretability by allowing the analysis to focus on major disease categories rather than highly specific and potentially rare diagnoses.

```python
diabetes['readmitted'].value_counts()
```

```
readmitted
NO     53821
>30    35173
<30    11250
Name: count, dtype: int64
```

The value count check for the readmitted column reveals a clear class imbalance in the dataset. The majority of records fall into the NO category (53,821 instances), followed by

>30 days (35,173 instances) and <30 days (11,250 instances). This imbalance indicates that the dataset is skewed toward non-readmission cases, which could bias the model toward predicting the majority class. Identifying this distribution is important because it highlights the need for techniques such as resampling (e.g., SMOTENC) or class weighting to ensure the model can learn effectively from all classes, especially the minority <30 days group that is often of highest clinical importance.

4. Building the classification model

4.1. Python

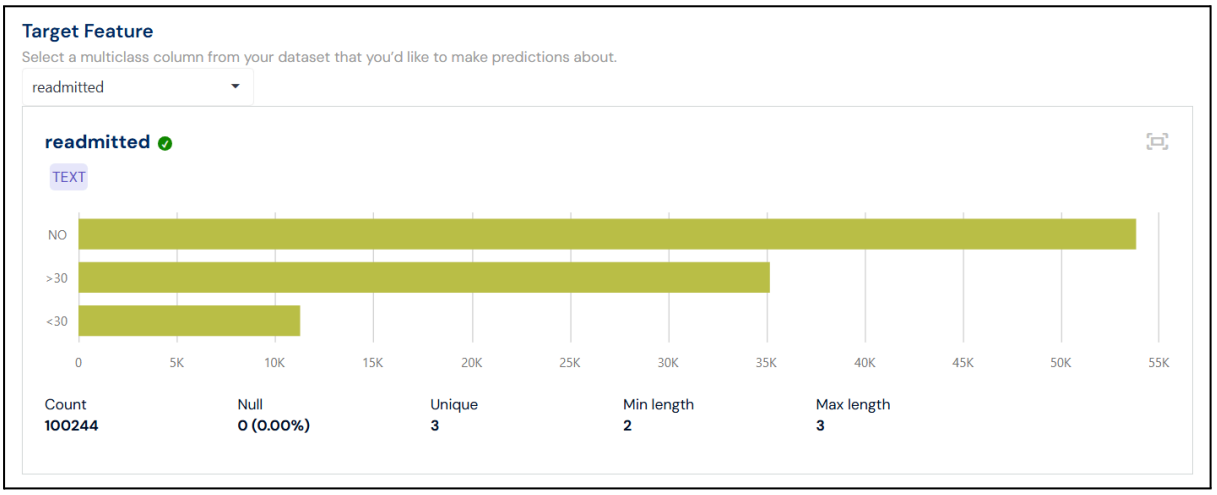
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)
```

```python
from imblearn.over_sampling import SMOTENC
# 1) Tên các cột categorical (chính là các cột bạn đã label-encode)
cat_cols = [col for col in X_train.columns if X_train[col].nunique() <= 20]
cat_idx = [X_train.columns.get_loc(c) for c in cat_cols]
# SMOTENC
smote = SMOTENC(categorical_features=cat_idx, random_state=42)
X_res, y_res = smote.fit_resample(X_train, y_train)
```

```python
from sklearn.ensemble import RandomForestClassifier
# Train model
model = RandomForestClassifier(random_state=42)
model.fit(X_res, y_res)
```

The predictive model used in this study is a Random Forest Classifier with a fixed random_state of 42 to ensure reproducibility. The dataset was split into training and testing sets in an 80–20 ratio, with stratification applied to preserve the original class distribution of the target variable. Given the presence of significant class imbalance, the SMOTENC (Synthetic Minority Over-sampling Technique for Nominal and Continuous variables) method was applied exclusively to the training set. This approach identifies categorical columns, defined as those with 20 or fewer unique values, and generates synthetic samples for the minority classes while respecting the discrete nature of categorical features. The resampled training set (X_res, y_res) was then used to fit the Random Forest model, ensuring more balanced learning across all classes.

4.2. Graphite Note



The target feature "readmitted" contains 100,244 records with no missing values (0.00% null). It is a multiclass variable with three unique categories: "NO", ">30", and "<30", representing whether a patient was not readmitted, readmitted after more than 30 days, or readmitted within 30 days, respectively. The distribution shows that the majority of patients fall into the "NO" category, followed by ">30", while "<30" has the smallest proportion. The text length for category labels ranges from a minimum of 2 characters to a maximum of 3 characters. This indicates an imbalanced target variable, which may influence model performance and require consideration during training.

5. Confusion Matrix result

5.1. Python

```
from sklearn.metrics import classification_report, confusion_matrix
y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 344  977  929]
 [ 808 3149 3078]
 [ 949 2928 6887]]
              precision    recall  f1-score   support

           0       0.16      0.15      0.16      2250
           1       0.45      0.45      0.45      7035
           2       0.63      0.64      0.64     10764

    accuracy                           0.52     20049
   macro avg       0.41      0.41      0.41     20049
weighted avg       0.51      0.52      0.52     20049
```

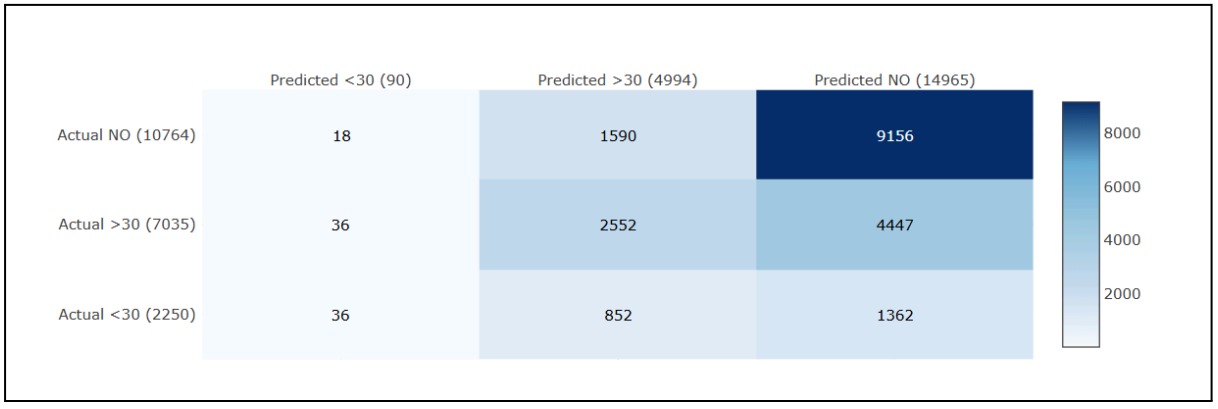
The model's performance analysis reveals several key insights. The overall accuracy of 52% indicates that the model is correctly classifying just over half of the instances, suggesting there is room for improvement. Precision is relatively higher for class 2 (0.63), indicating the model is better at identifying positive instances for this class, but it struggles with class 0 (0.16) and class 1 (0.45), where it misclassifies many instances. Recall follows a similar pattern, with class 2 performing best (0.64), but class 0 (0.15) and class 1 (0.45) show significant challenges in capturing actual positive instances. The F1 score, which balances precision and recall, is strongest for class 2 (0.64), but much lower for classes 0 and 1 (0.16 and 0.45), emphasizing the difficulty in achieving balanced performance across all classes.

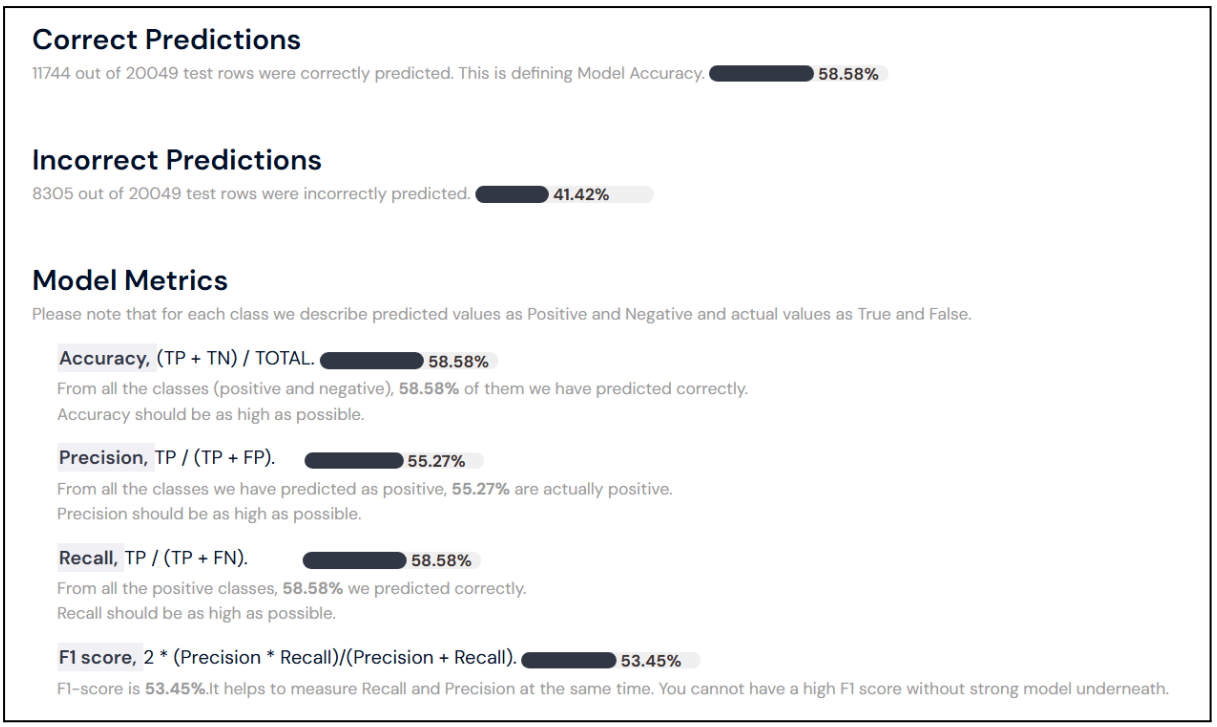
```
from sklearn.metrics import roc_auc_score
y_proba = model.predict_proba(X_test)
print("ROC-AUC:", roc_auc_score(y_test, y_proba, multi_class='ovr'))
```

```
ROC-AUC: 0.6242816741399592
```

The ROC-AUC of 0.624 suggests moderate discrimination ability, but the model still has considerable room for improvement in distinguishing between the classes. Overall, while the model performs adequately in some areas, particularly for class 2, there are significant opportunities to enhance its precision, recall, and F1 scores, especially for the underperforming classes.cing techniques could help enhance the model's discriminative performance.

5.2. Graphite Note





The model's overall performance in predicting the "readmitted" target feature is moderately effective, with the following key metrics: Accuracy of 58.58%, Precision of 55.27%, Recall of 58.58%, F1 score of 53.45%, and an AUC of 66.76%. The accuracy and recall values suggest the model is relatively balanced in its predictions, but there is room for improvement, especially in precision and F1 score. The AUC indicates a fair ability to distinguish between the classes, but further tuning may be required to optimize performance across all metrics. Overall, the model shows potential but could benefit from additional refinement to enhance its prediction capabilities.

6. Domain Comparison

		Python	Graphite Note
Class 0 (<30 day)	Precision	0.16	0.4
	Recall	0.15	0.016
	F1-score	0.16	0.031
		Python	Graphite Note
Class 1 (>30 days)	Precision	0.45	0.511
	Recall	0.45	0.363
	F1-score	0.45	0.422
		Python	Graphite Note
Class 2 (No readmission)	Precision	0.63	0.577
	Recall	0.64	0.851
	F1-score	0.64	0.688
		Python	Graphite Note
	Accuracy	0.52	0.586
	AUC	0.62	0.669

Overall, the model performance results from Python and Graphite Note show noticeable differences across classes and metrics. For Class 0 (<30 days), Graphite Note reports much higher precision (0.4 vs. 0.16) but a significantly lower recall (0.016 vs. 0.15), resulting in a much lower F1-score compared to Python. For Class 1 (>30 days), Graphite Note achieves slightly higher precision (0.511 vs. 0.45) but lower recall (0.363 vs. 0.45), leading to a marginally lower F1-score. In Class 2 (No readmission), Python produces slightly better precision (0.63 vs. 0.577), whereas Graphite Note achieves much higher recall (0.851 vs. 0.64), resulting in a higher F1-score. When looking at overall performance, Graphite Note outperforms Python in accuracy (0.586 vs. 0.52) and AUC (0.669 vs. 0.62), indicating better overall classification capability despite mixed results across individual classes.

7. Insights for Hospital Business in the U.S.

To reduce costly readmissions and improve patient care, the hospital should prioritize a targeted intervention program for high-risk patients, especially those likely to be readmitted within 30 days. This can include enhanced discharge planning, personalized follow-up schedules, and telehealth check-ins to address post-discharge complications early. At the same time, establishing a chronic care management pathway for patients with conditions leading to longer-term readmissions can help maintain engagement and adherence to treatment plans. Additionally, strengthening preventive care programs for patients with low readmission risk will sustain their healthy status while optimizing resource allocation. Collaboration between medical teams, data analysts, and case managers will be essential to continuously refine patient segmentation, enabling the hospital to improve outcomes, reduce penalty costs from insurers or government health programs, and enhance its reputation for quality care.

8. Future improvements

To improve the model's performance, we can try a few strategies. Further improvements can be made by trying more advanced models like Gradient Boosting or XGBoost, which often handle imbalanced data better. Tuning hyperparameters with tools like GridSearchCV or RandomizedSearchCV can optimize model performance, while adding or transforming features (such as scaling, polynomial features, or interaction terms) could also improve results. Consider exploring ensemble methods like Balanced Random Forest or

EasyEnsemble, which are designed for imbalanced datasets. Shifting focus from accuracy to metrics like precision, recall, and F1 score will provide more meaningful insights, especially for the minority class. Additionally, using Stratified K-Fold Cross Validation will ensure that each fold maintains a similar class distribution, providing more reliable performance estimates. These steps, combined with SMOTENC, should lead to better model accuracy and a more balanced prediction.

Link to Github