EIU
TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

15 Year Anniversary 2010 - 2025

Business Applications of Predictive Analytics with AI-Based Tools

COURSE: MIS 395 _ Artificial Intelligence for Business

LECTURE: Mr. Dang Thai Doan

PRESENTED BY: Group 3A

TEAM MEMBERS

Bùi Thị Thanh Thảo
2232300157

Nguyễn Thành Đạt
2132300562

Lê Nguyễn Tâm Như
2132300065

TABLE OF CONTENT

DATASET 1

Superstore Sales Data

DATASET 2

California Housing Prices

DATASET 3

Diabetes 130 US hospitals for years 1999-2008

I. Dataset: Superstore Sales Data

Data Overview

Topic: Sales transactions of a superstore, including order dates, products, customers information and their expenditure.

- **Rows**: 2,240, **columns**: 22
- **The goal:** To analyze sales performance and predict customer spending in the future.
- **Model type**: Regression (for spending prediction).
- **Application**: Used in retail business analysis, performance reporting, and strategic planning.

I. Dataset: Superstore Sales Data

Data Preprocessing

Date conversion & feature creation: Converted Dt_Customer to datetime format; calculated Age; created Total_Spending from six product categories.

Normalization: Applied MinMaxScaler to normalize Income and Total_Spending to the [0,1] range for balanced weighting.

Data cleaning: Removed unrealistic records (Year_Birth < 1920 or Age ⩾ 100).

Category grouping: Grouped rare values in Marital_Status (“YOLO”, “Absurd”) into the “Other” category.

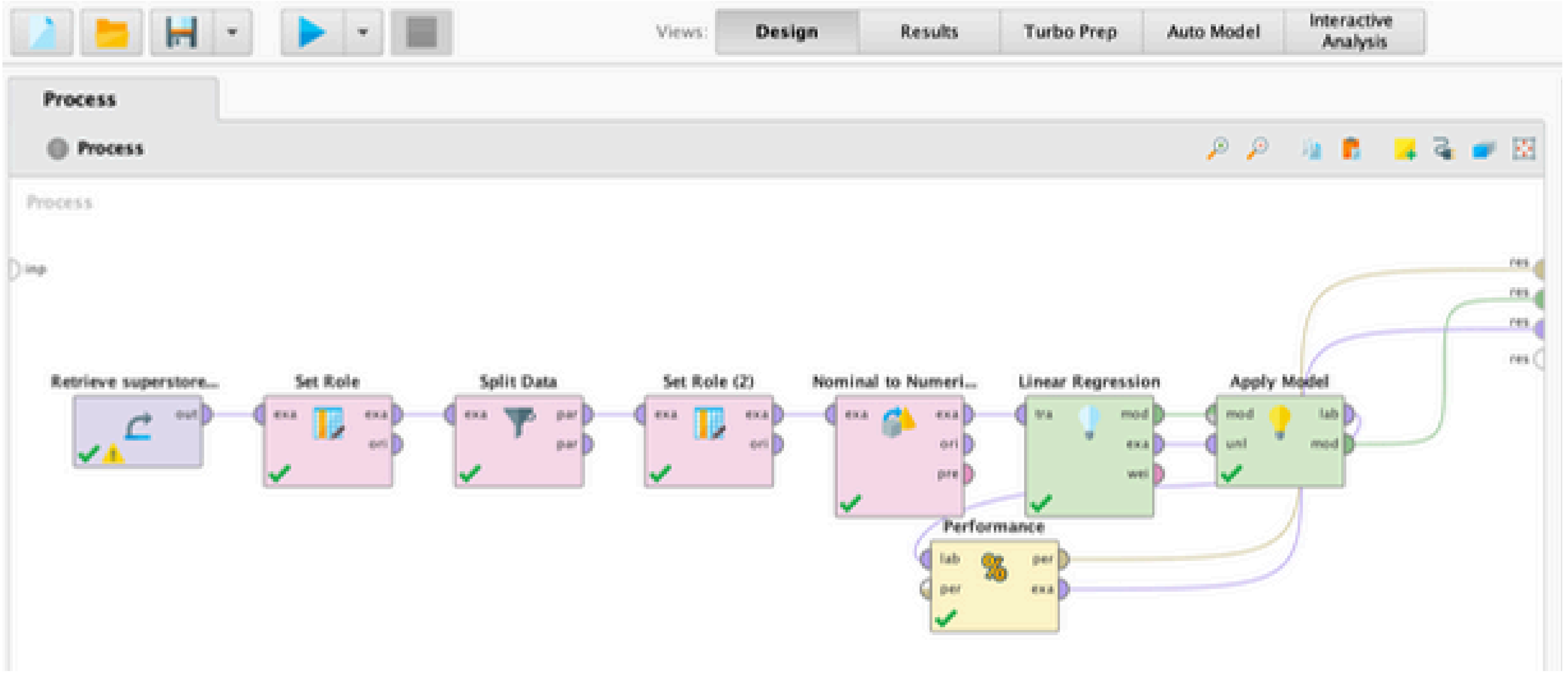
⟶ Finally, the cleaned and transformed dataset was saved locally as a CSV file for further analysis using RapidMiner.

I. Dataset: Superstore Sales Data

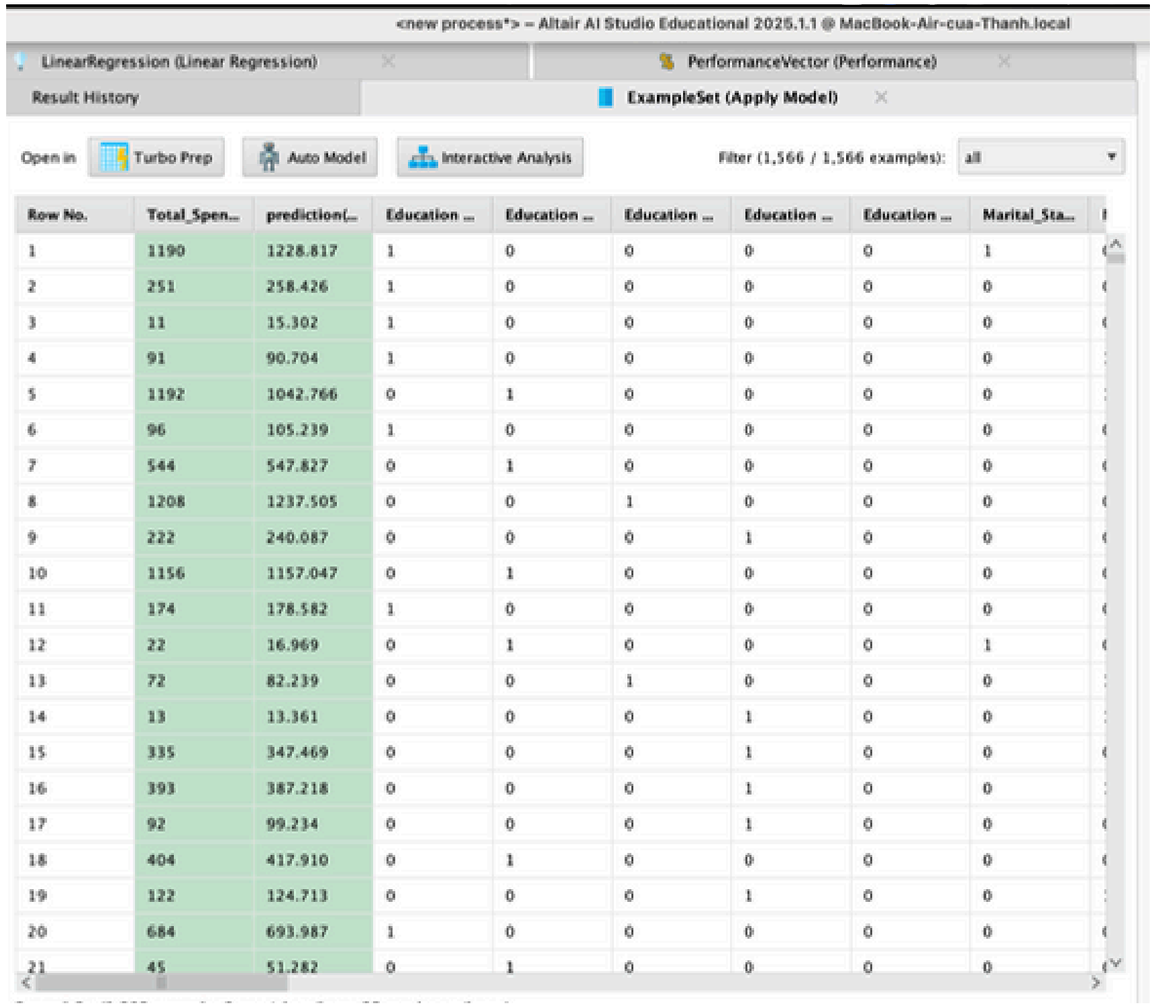
Descriptive Analytics

- Age vs. Spending: Middle-aged customers (40–60) are the highest spenders
- Marital Status: Married and “Together” customers spend more.
- Campaign Response: Respondents to marketing campaigns spend more.
- Product Category: Wine is the top-purchased product.

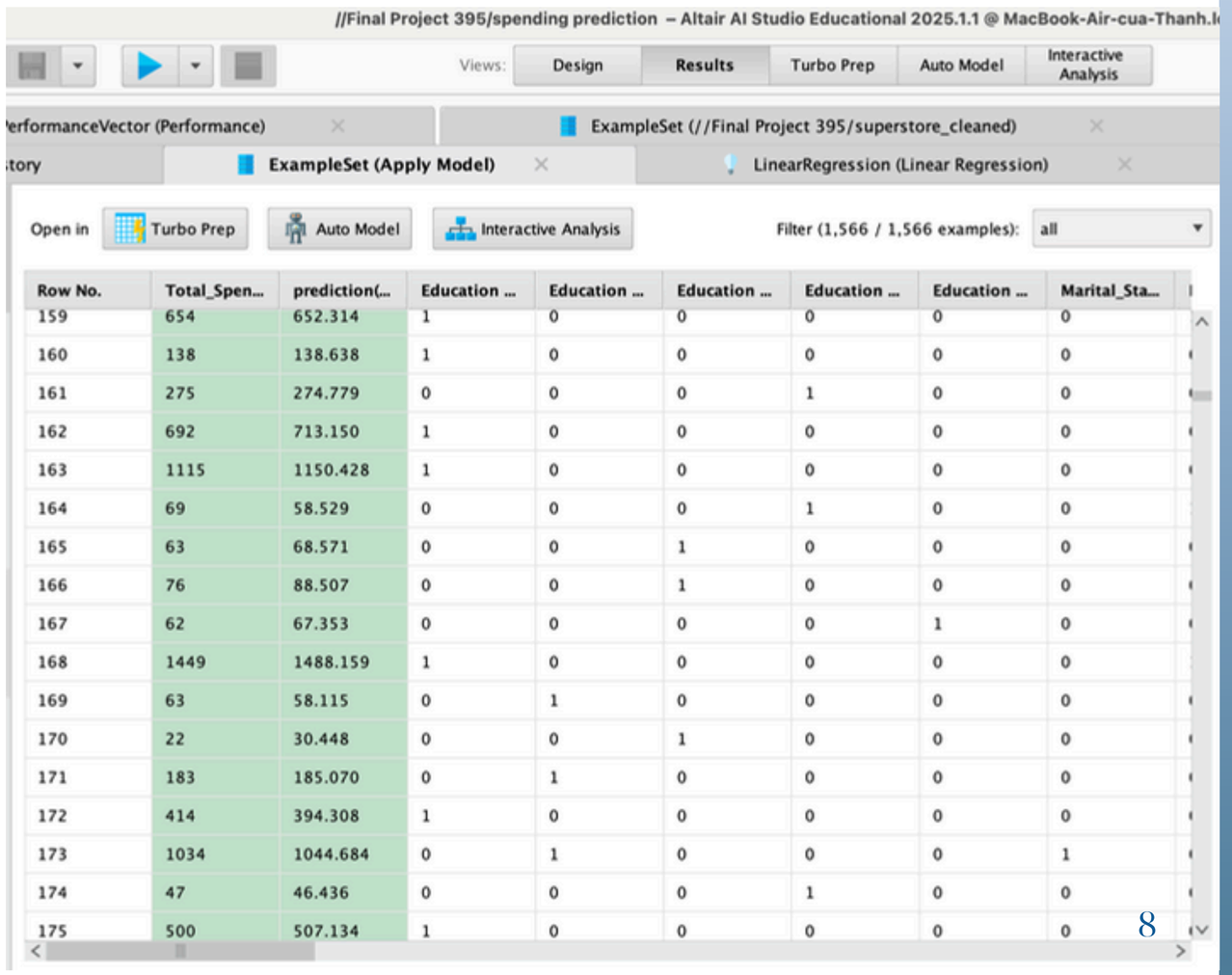
I. Dataset: Superstore Sales Data



I. Dataset: Superstore Sales Data



Row No.	Total_Spen...	prediction(...	Education ...	Education ...	Education ...	Education ...	Education ...	Marital_Sta...
1	1190	1228.817	1	0	0	0	0	1
2	251	258.426	1	0	0	0	0	0
3	11	15.302	1	0	0	0	0	0
4	91	90.704	1	0	0	0	0	0
5	1192	1042.766	0	1	0	0	0	0
6	96	105.239	1	0	0	0	0	0
7	544	547.827	0	1	0	0	0	0
8	1208	1237.505	0	0	1	0	0	0
9	222	240.087	0	0	0	1	0	0
10	1156	1157.047	0	1	0	0	0	0
11	174	178.582	1	0	0	0	0	0
12	22	16.969	0	1	0	0	0	1
13	72	82.239	0	0	1	0	0	0
14	13	13.361	0	0	0	1	0	0
15	335	347.469	0	0	0	1	0	0
16	393	387.218	0	0	0	1	0	0
17	92	99.234	0	0	0	1	0	0
18	404	417.910	0	1	0	0	0	0
19	122	124.713	0	0	0	1	0	0
20	684	693.987	1	0	0	0	0	0
21	45	51.282	0	1	0	0	0	0



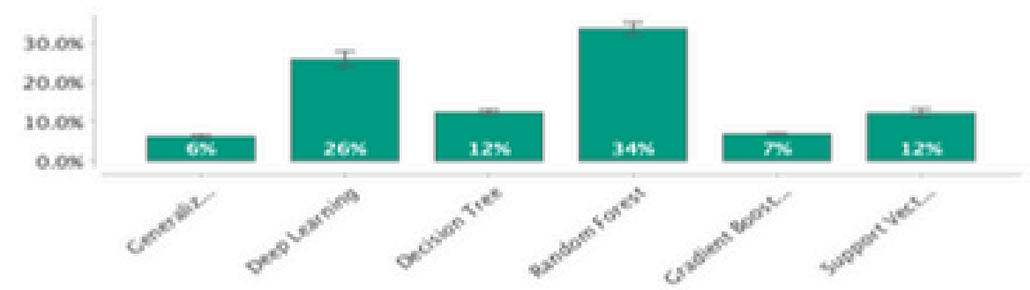
Row No.	Total_Spen...	prediction(...	Education ...	Education ...	Education ...	Education ...	Education ...	Marital_Sta...
159	654	652.314	1	0	0	0	0	0
160	138	138.638	1	0	0	0	0	0
161	275	274.779	0	0	0	1	0	0
162	692	713.150	1	0	0	0	0	0
163	1115	1150.428	1	0	0	0	0	0
164	69	58.529	0	0	0	1	0	0
165	63	68.571	0	0	1	0	0	0
166	76	88.507	0	0	1	0	0	0
167	62	67.353	0	0	0	0	1	0
168	1449	1488.159	1	0	0	0	0	0
169	63	58.115	0	1	0	0	0	0
170	22	30.448	0	0	1	0	0	0
171	183	185.070	0	1	0	0	0	0
172	414	394.308	1	0	0	0	0	0
173	1034	1044.684	0	1	0	0	0	1
174	47	46.436	0	0	0	1	0	0
175	500	507.134	1	0	0	0	0	0

I. Dataset: Superstore Sales Data

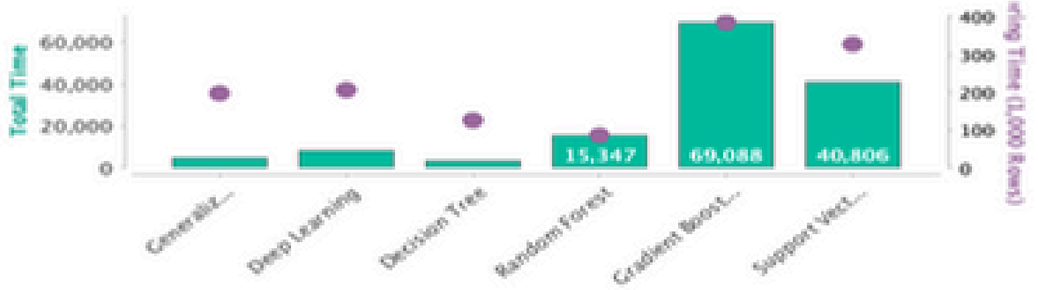
Overview

Number of Models: 8

Relative Error



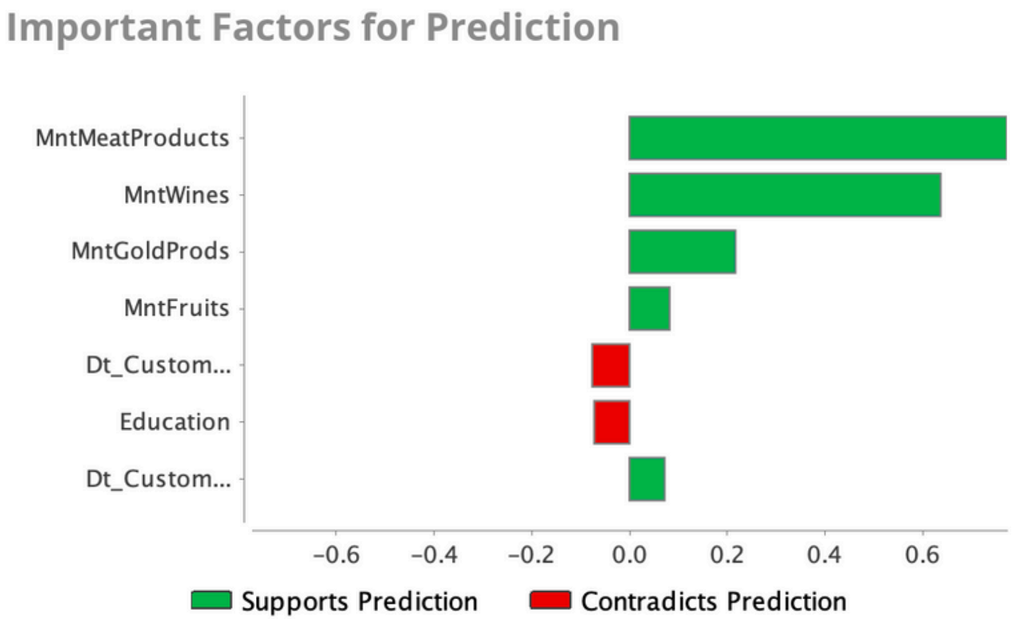
Runtimes (ms)

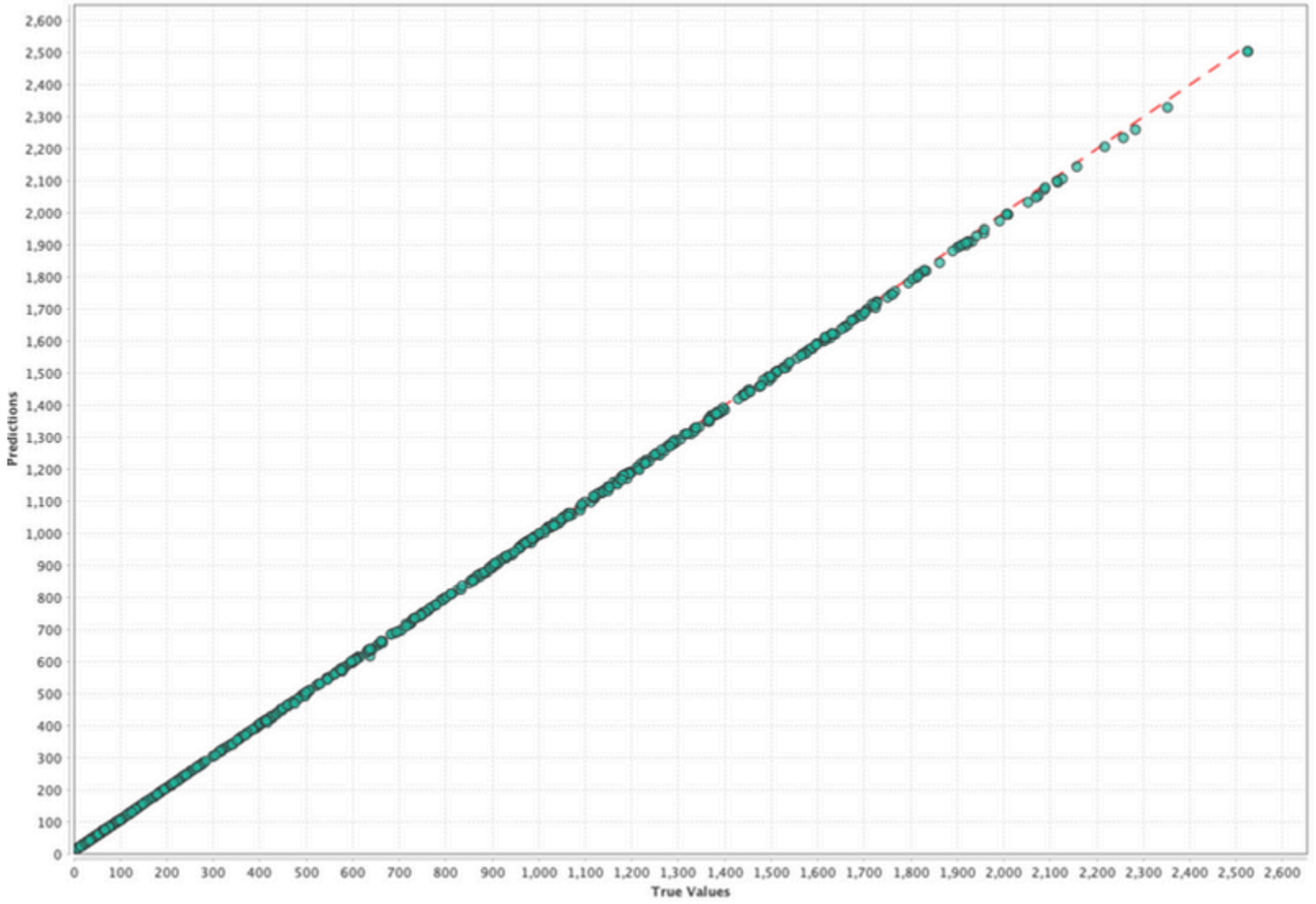


Relative Error

Model	Relative Error	Standard Deviation	Gains	Total Time	Training
Generalized Linear Model	6.2%	± 0.5%	?	5 s	100 ms
Deep Learning	25.9%	± 2.0%	?	8 s	834 ms
Decision Tree	12.4%	± 0.6%	?	3 s	46 ms
Random Forest	33.7%	± 1.6%	?	15 s	131 ms
Gradient Boosted Trees	6.8%	± 0.2%	?	1 min 9 s	1 s

I. Dataset: Superstore Sales Data





I. Dataset: Superstore Sales Data

	Value (± Std. Dev.)	Micro Average (± Std. Dev.)
Absolute Error	6.372 ± 0.296	6.373 ± 3.942
Root Mean Squared Error (RMSE)	7.480 ± 0.489	7.493 ± 0.000
Squared Error (MSE)	56.144 ± 7.319	56.152 ± 74.719
Correlation	1.000 ± 0.000	1.000 ± 0.000

The model demonstrates very high accuracy with minimal prediction errors (AE ≈ 6.37, RMSE ≈ 7.48).

Perfect correlation (1.0) indicates predictions align almost exactly with actual spending.

Model performance is strong and reliable for forecasting customer spending behavior.

I. Dataset: Superstore Sales Data

Business Insights

- Target **40–60-year-old customers** with moderate to high incomes.
- Focus **promotions** on **wine and meat** as top revenue categories.
- Use **past campaign response** history for **predictive targeting**.
- Integrate best-performing model **(GLM) into CRM** for spending forecasts, campaigns, and optimized segmentation.
- Upselling & Cross-selling: Offer **bundle deals** (e.g., wine + meat) to increase basket size.
- Customer Loyalty Programs: Reward high spenders and campaign responders to **boost retention**.
- **Limitation:** Lacks temporal and geographic data; future work should add transaction timelines and customer feedback for deeper insights.

II. Dataset California Housing Prices

Dataset Overview

- **Dataset:** California Housing Prices
- **Size**: 20,640 rows, 10 columns
- **Key Features:** Median house value, number of rooms, geographical data (latitude, longitude), Housing_median_age, total_rooms, total_bedrooms, population, households, median_income
- **Goal**: Predict housing prices based on features such as location, income, and household characteristics.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41	880	129.0	322	126	8.3252	452600	NEAR BAY
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	358500	NEAR BAY
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	352100	NEAR BAY
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	341300	NEAR BAY
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	342200	NEAR BAY

II. Dataset: California Housing Prices

Data Preprocessing & Cleaning

Handling Missing Values

Mean imputation for total_bedrooms based on ocean proximity

```
house_price.isna().sum()

longitude                0
latitude                 0
housing_median_age       0
total_rooms              0
total_bedrooms         207
population               0
households               0
median_income            0
median_house_value       0
ocean_proximity          0
dtype: int64
```

Normalization

Features like ‘total_rooms', 'total_bedrooms', 'population', 'households', 'median_income' (mean = 0, std = 1)

```
from sklearn.preprocessing import StandardScaler

columns_to_scale = ['total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']
# Khởi tạo StandardScaler
scaler = StandardScaler()
# Chuẩn hóa dữ liệu
house_price[columns_to_scale] = scaler.fit_transform(house_price[columns_to_scale])
# Kiểm tra lại dữ liệu đã được chuẩn hóa
print(house_price[columns_to_scale].head())
```

```
   total_rooms  total_bedrooms  population  households  median_income
0    -0.804819       -0.975250   -0.974429   -0.977033       2.344766
1     2.045890        1.355060    0.861439    1.669961       2.332238
2    -0.535746       -0.829755   -0.820777   -0.843637       1.782699
3    -0.624215       -0.722422   -0.766028   -0.733781       0.932968
4    -0.462404       -0.615089   -0.759847   -0.629157      -0.012881
```

II. Dataset: California Housing Prices

Data Preprocessing & Cleaning

Data Split

80% training and 20% testing

```
X_all = house_price.drop(columns=["median_house_value"])  # Drop only the target variable
y_all = house_price['median_house_value']

X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(X_all, y_all, test_size=0.2, random_state=42)

model_all = LinearRegression()
model_all.fit(X_train_all, y_train_all)
```

```
LinearRegression()
```

```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=42)
```

Feature Transformation

One-hot encoding for ocean_proximity

```
house_price = pd.get_dummies(house_price, columns=['ocean_proximity'], drop_first=True)
```

```
house_price.info()
```

```
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   longitude                   20640 non-null  float64
 1   latitude                    20640 non-null  float64
 2   housing_median_age          20640 non-null  int64
 3   total_rooms                 20640 non-null  int64
 4   total_bedrooms              20640 non-null  float64
 5   population                  20640 non-null  int64
 6   households                  20640 non-null  int64
 7   median_income               20640 non-null  float64
 8   median_house_value          20640 non-null  int64
 9   ocean_proximity_INLAND      20640 non-null  bool
 10  ocean_proximity_ISLAND      20640 non-null  bool
 11  ocean_proximity_NEAR BAY    20640 non-null  bool
 12  ocean_proximity_NEAR OCEAN  20640 non-null  bool
```

II. Dataset: California Housing Prices

Exploratory Data Analysis (EDA)

Descriptive Statistics

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

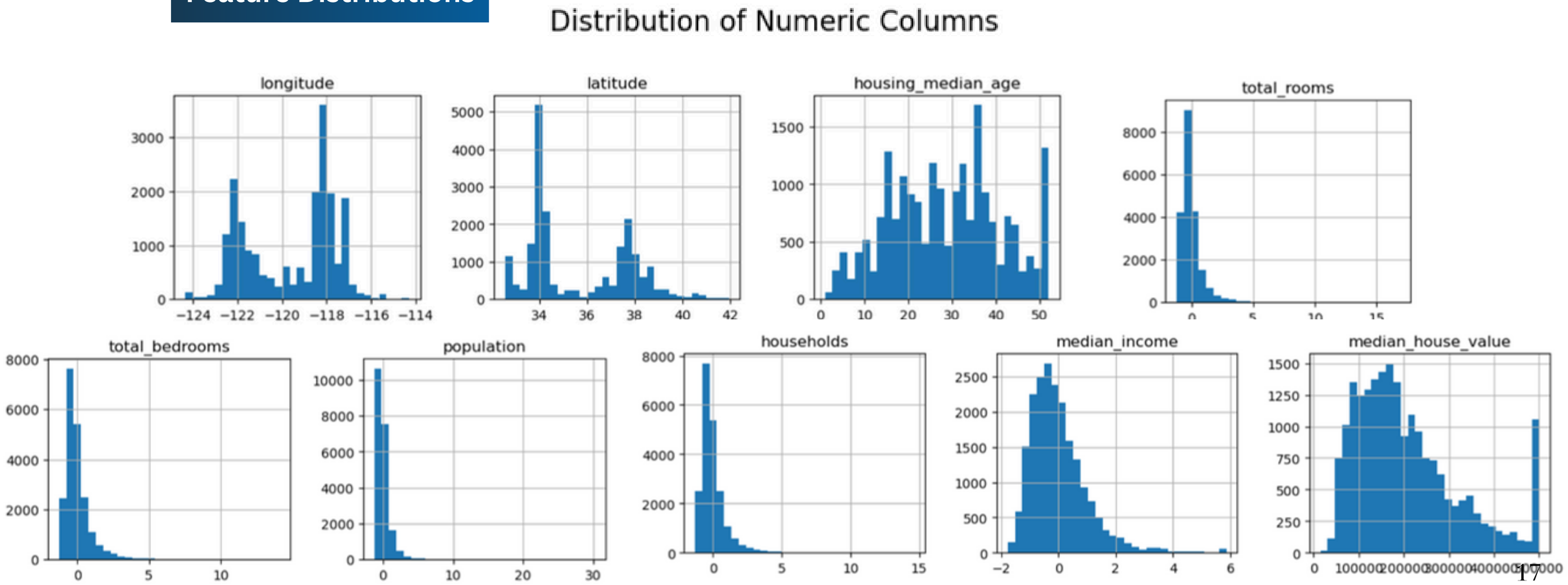
After Normalization

	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

II. Dataset: California Housing Prices

Exploratory Data Analysis (EDA)

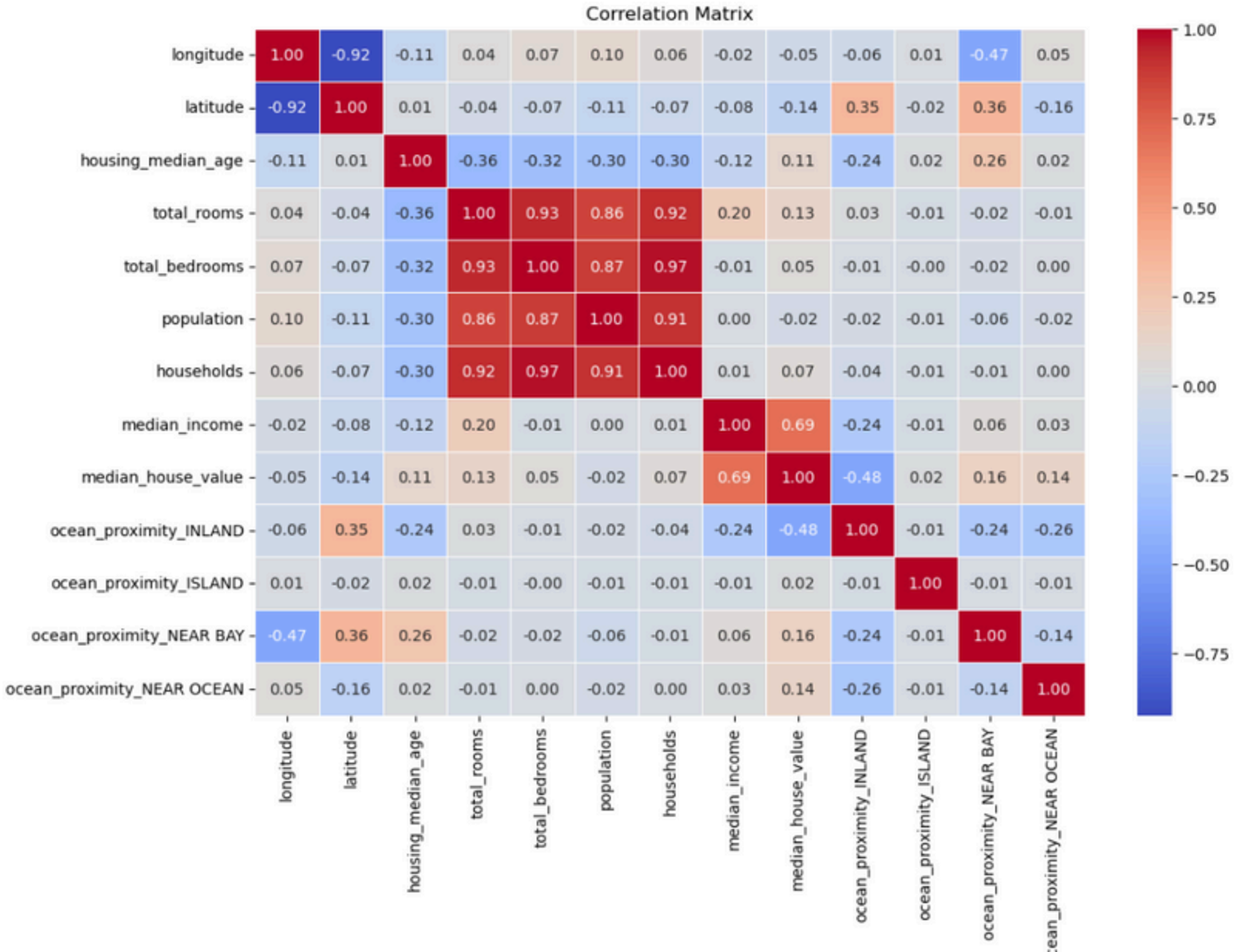
Feature Distributions



II. Dataset: California Housing Prices

Exploratory Data Analysis (EDA)

Correlation

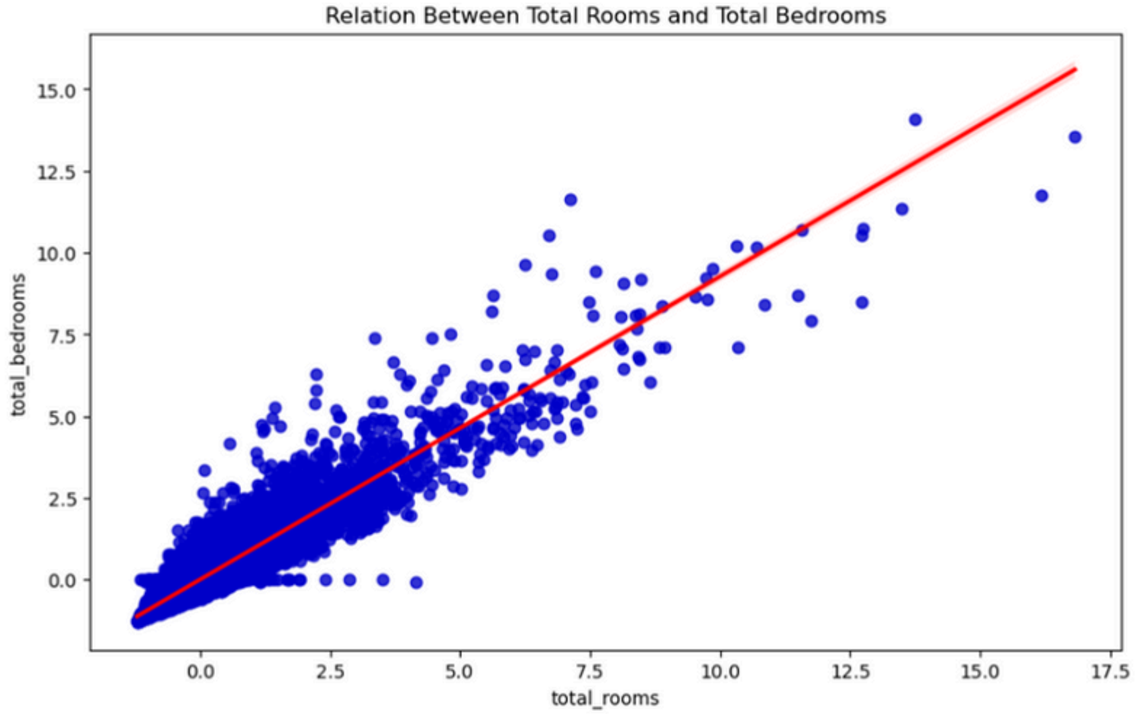


II. Dataset: California Housing Prices

Exploratory Data Analysis (EDA)

Scatter Plots

Total_rooms vs Total_bedrooms: Strong positive correlation, showing more rooms tend to mean more bedrooms.



Median_income vs Median_house_value: Clear upward trend, indicating higher income leads to higher home prices.

II. Dataset: California Housing Prices

Exploratory Data Analysis (EDA)

Linear Regression

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

Clearer interpretation, slightly lower accuracy.

Random Forest

MAE	31,670.37
MSE	24,036,753,19.9
R-squared	~ 0.82

High accuracy, captures complex patterns well

Both models produced similar results, with **Random Forest** being more robust in capturing non-linear relationships.

II. Dataset: California Housing Prices

Tool Comparison (Jupyter vs Deepnote)

Jupyter Notebook

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

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

Deepnote

- Mean Absolute Error (MAE): 31670.37
- Mean Squared Error (MSE): 2403675319.9
- R-squared value: 0.8166

- Mean Absolute Error (MAE): 50704.38
- Mean Squared Error (MSE): 4904902402.99
- R-squared value: 0.6257

Both tools produced similar accuracy
Deepnote was faster due to AI support

II. Dataset: California Housing Prices

Insights for Business

- **Median Income:** Strong correlation with higher property values, suggesting higher-income areas have more expensive homes.
- **Total Rooms/Bedrooms:** Homes with more rooms and bedrooms tend to have higher values, guiding pricing strategies.
- **Ocean Proximity:** Homes near the ocean are more expensive. Proximity to key features like the ocean and income levels are significant predictors of property prices.
- **Business Application:** Insights help in pricing, market segmentation, and identifying high-potential investment areas.

III. Diabetes 130 US hospitals (1999-2008)

Dataset Context

The dataset includes ten years (1999-2008) of clinical data from **130 hospitals in the US**, covering diabetic patients who underwent lab tests, received medications, and stayed up to 14 days. It seeks to **predict early readmission within 30 days after discharge**. Although proper diabetes care improves patient outcomes, many patients do not receive it, resulting in higher hospital costs, more readmissions, and increased health risks.

III. Diabetes

Dataset Preprocessing

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	...	citoglipton
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
5	35754	82637451	Caucasian	Male	[50-60)	?	2	1	2	3	...	No

III. Diabetes

Dataset Preprocessing

Attribute	Missing Rows
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

Decision:

- I removed all attributes that have large percent of missing values, included race.
- Then I dropped missing rows from diag_1 to diag_3 in order to avoid the influence into our model.

→ After processing missing values, the data has 100244 rows and 42 attributes.

III. Diabetes

Dataset Preprocessing

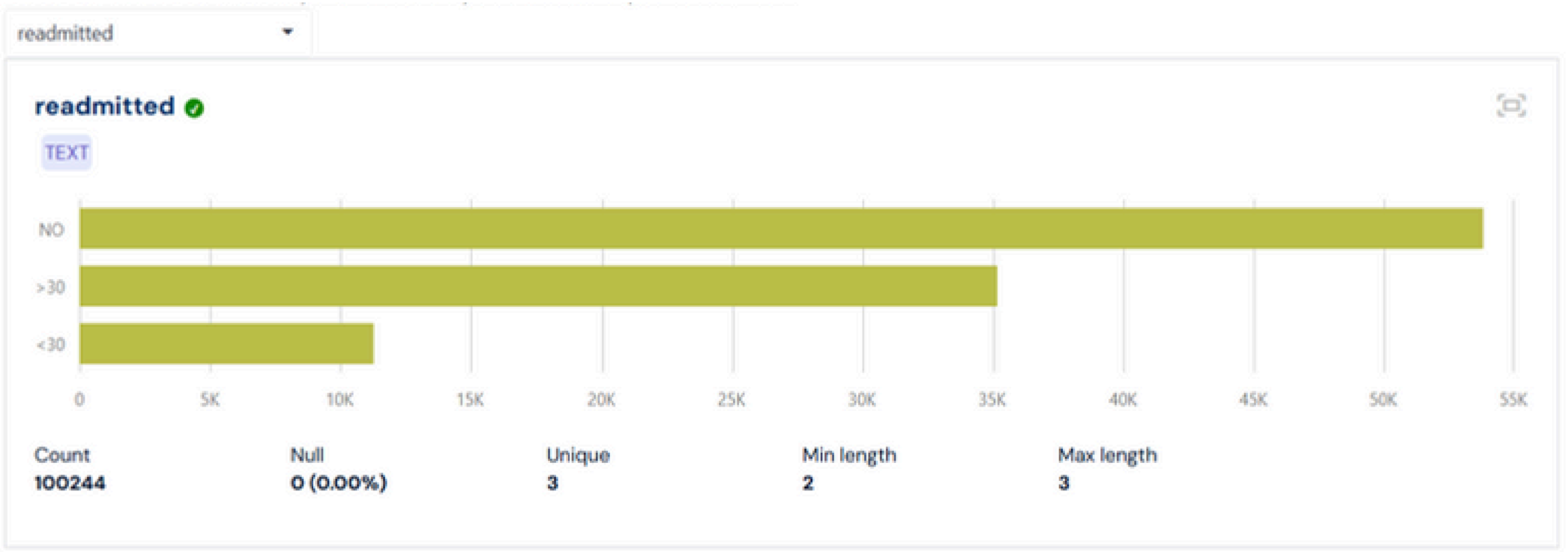
ICD-9 (International Classification of Diseases, 9th Edition) is a system used to classify and code a wide variety of diseases, conditions, and procedures. It was developed by the World Health Organization (WHO) and used worldwide for medical billing, epidemiological studies, and research.

```
if 1 <= num <= 139:
    return 'Infectious'
elif 140 <= num <= 239:
    return 'Neoplasms'
elif 240 <= num <= 279:
    return 'Endocrine'
elif 280 <= num <= 289:
    return 'Blood'
elif 290 <= num <= 319:
    return 'Mental'
elif 320 <= num <= 389:
    return 'Nervous'
elif 390 <= num <= 459:
    return 'Circulatory'
elif 460 <= num <= 519:
    return 'Respiratory'
```

```
elif 520 <= num <= 579:
    return 'Digestive'
elif 580 <= num <= 629:
    return 'Genitourinary'
elif 630 <= num <= 679:
    return 'Pregnancy'
elif 680 <= num <= 709:
    return 'Skin'
elif 710 <= num <= 739:
    return 'Musculoskeletal'
elif 740 <= num <= 759:
    return 'Congenital'
elif 760 <= num <= 779:
    return 'Perinatal'
elif 780 <= num <= 799:
    return 'Symptoms'
elif 800 <= num <= 999:
    return 'Injury'
else:
    return 'Unknown'
```

III. Diabetes

Target Feature Distribution



III. Diabetes

Dataset Preprocessing

Most of the values in several columns were originally categorical but had been mapped from text labels to numeric codes. To ensure the predictive model interprets these variables correctly as categorical features rather than continuous numerical values, a reverse mapping process was applied to convert the numeric codes back to their original categorical labels. This step preserved the semantic meaning of the data and prevented the model from making false assumptions about ordinal relationships that do not actually exist.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in diabetes:
    diabetes[col] = le.fit_transform(diabetes[col])
```

III. Diabetes

Model Building in Python

The next stage involves splitting the data into training and testing subsets using train_test_split from sklearn.model_selection. A test size of 20% is specified, and stratify=y ensures that the class distribution is preserved in both sets, which is particularly important when working with imbalanced target variables. A fixed random_state is used to make the split reproducible.

```
X = diabetes.drop('readmitted', axis=1)
y = diabetes['readmitted']
```

```
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)
```

III. Diabetes

Model Building in Python

To address the class imbalance identified earlier, the SMOTENC technique is implemented. SMOTENC creates synthetic samples for the minority classes, generating numeric features through interpolation and categorical features by selecting the most frequent category among nearest neighbors. This results in a balanced training dataset without distorting categorical variables. Finally, the RandomForestClassifier from sklearn.ensemble is imported in preparation for model training. This algorithm is a powerful and flexible ensemble learning method based on decision trees.

```
from imblearn.over_sampling import SMOTENC

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]
smote = SMOTENC(categorical_features=cat_idx, random_state=42)
X_res, y_res = smote.fit_resample(X_train, y_train)
```

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_res, y_res)
```

III. Diabetes

Model Building in Python

To address the class imbalance identified earlier, the SMOTENC technique is implemented. First, the categorical columns are identified as those with 20 or fewer unique values. Their positions are stored in cat_idx, and passed to SMOTENC so that the algorithm knows which features require categorical handling. SMOTENC creates synthetic samples for the minority classes, generating numeric features through interpolation and categorical features by selecting the most frequent category among nearest neighbors. This results in a balanced training dataset without distorting categorical variables.

```
from imblearn.over_sampling import SMOTENC

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]
smote = SMOTENC(categorical_features=cat_idx, random_state=42)
X_res, y_res = smote.fit_resample(X_train, y_train)
```

III. Diabetes

Model Building in Python

Class 0 (<30 days):
- Precision: 0.16
- Recall: 0.15
- F1-score: 0.16

Class 1 (>30 days):
- Precision: 0.45
- Recall: 0.45
- F1-score: 0.45

Class 2 (No readmission):
- Precision: 0.63
- Recall: 0.64
- F1-score: 0.64

Accuracy: 0.52

AUC: 0.62

Python			
	Predicted <30	Predicted >30	Predicted NO
Actual NO (10764)	949	2928	6887
Actual >30 (7035)	808	3149	3078
Actual <30 (2250)	344	977	929

III. Diabetes

Model Building in Graphite Note

Graphite Note			
	Predicted <30	Predicted >30	Predicted NO
Actual NO	18	1590	9156
Actual >30	36	2552	4447
Actual <30	36	852	1362

Class 0 (<30 days):
- Precision: 0.4
- Recall: 0.016
- F1-score: 0.031

Class 1 (>30 days):
- Precision: 0.511
- Recall: 0.363
- F1-score: 0.422

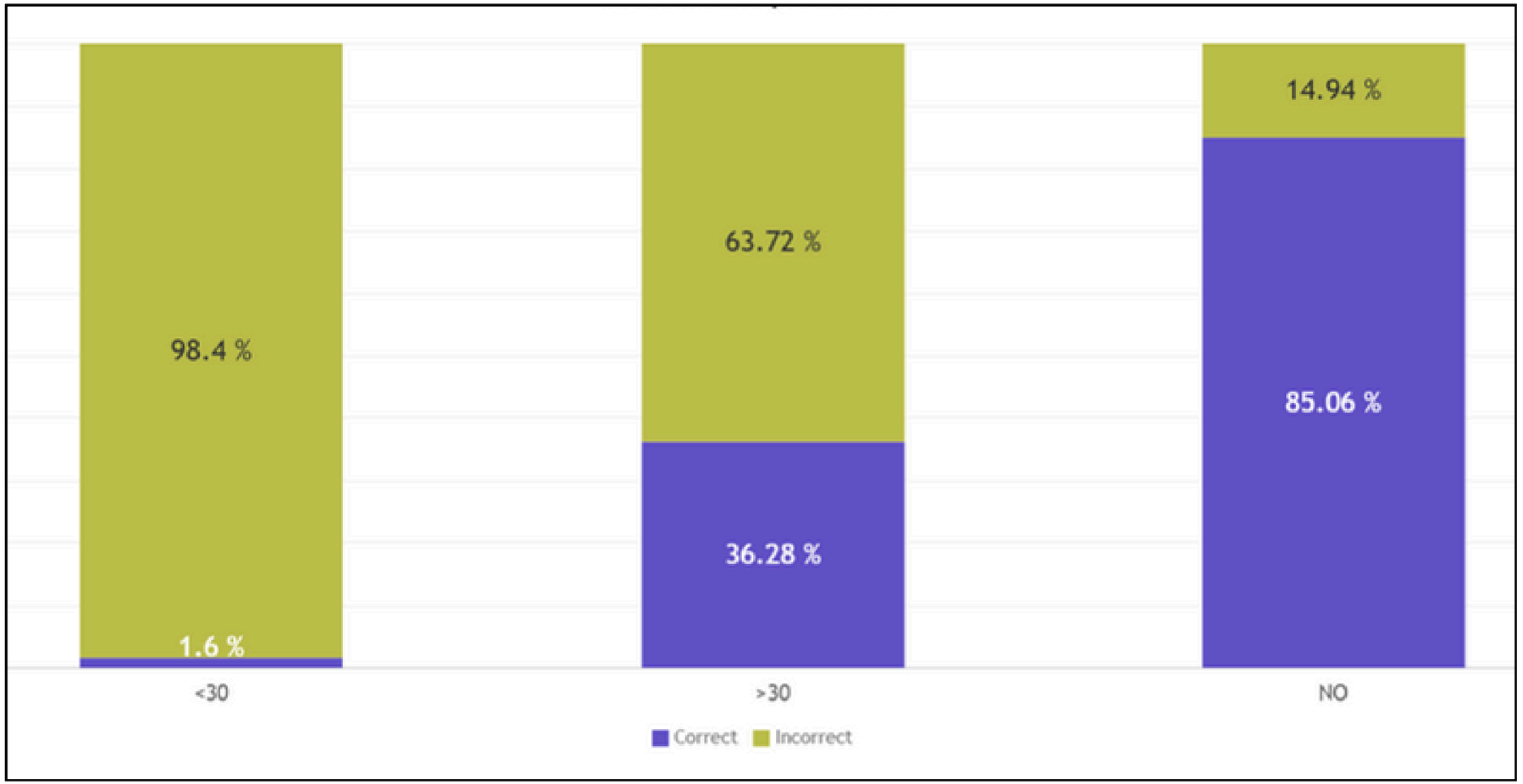
Class 2 (No readmission):
- Precision: 0.577
- Recall: 0.851
- F1-score: 0.688

Accuracy: 0.586

AUC: 0.669

III. Diabetes

Model Fit and Prediction on Graphite Note



III. Diabetes

Model Performance Comparison Between 2 Tools

		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

Thank You For Your Attention