Milestone 2 Report: Feature Engineering, Feature Selection, and Data Modeling

Sai Pande (37696687)

Objective:

The real estate market is a dynamic and multifaceted domain shaped by various economic, geographic, and social factors. In this project, I aim to analyze and predict real estate trends using historical housing data, enriched with visualization and modeling techniques. The goal is to uncover meaningful patterns in property prices, identify the key drivers of value, and provide actionable insights for buyers, sellers, and policymakers.

Real estate prices are influenced by features such as location, property size, number of rooms, year built, and nearby amenities. Through thorough preprocessing, exploratory data analysis, and machine learning models, this project evaluates how these attributes correlate with housing prices. By building predictive models and interactive dashboard, I strive to make complex real estate data more accessible and useful for decision-making.

Github link -> https://github.com/SaiPande/cap5771sp25-project

Datasets used:

Kaggle Dataset ->

Real Estate Sales 2001-2021 https://www.kaggle.com/datasets/utkarshx27/real-estate-sales-2001-2021-gl (License : CC0: Public Domain)

Zillow Economics Data https://www.kaggle.com/datasets/zillow/zecon (License : Data files © Original Authors)

Real Estate DataSet https://www.kaggle.com/datasets/arslanali4343/real-estate-dataset (License : CC0: Public Domain)

Tool Type:

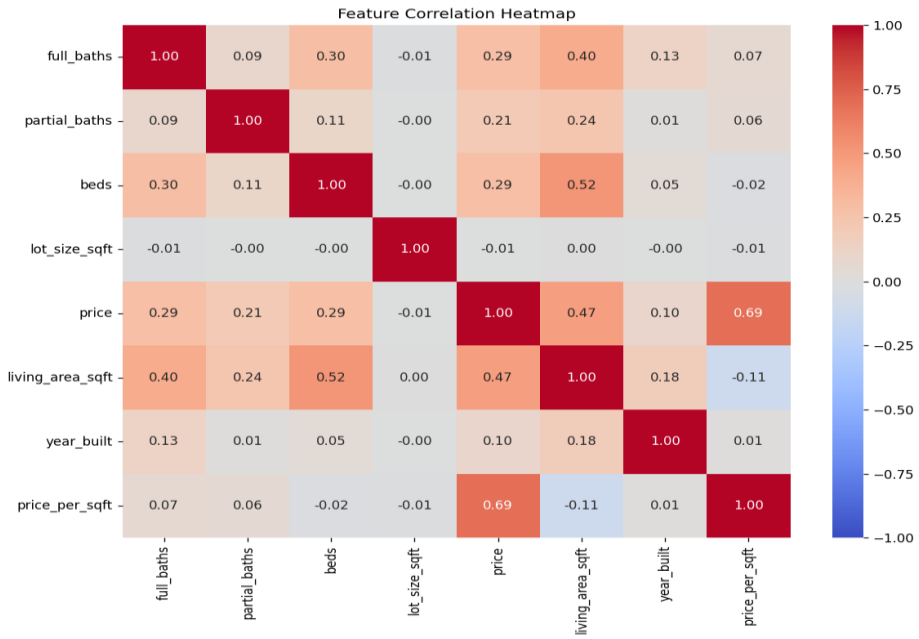
The project centers on building a predictive modeling tool. It includes both classification and regression models to forecast stock movement (direction and magnitude). In the final stage, this tool will be deployed as a conversational AI chatbot to allow users to query future stock predictions and receive responses grounded in macroeconomic and technical indicators. This interface will enhance interpretability and accessibility.

Exploratory Data Analysis (EDA) Recap:

EDA revealed important trends:

- Pacific time zone properties had notably higher prices.
- Features such as square footage, lot size, number of beds, and total bathrooms were positively correlated with price.
- Outliers in lot size and skewed distributions in features like price and area were handled using log transformations and IQR-based filtering.

EDA surfaced several issues such as skewed distributions, correlated variables, and city-level pricing extremes. These findings inform both feature engineering and model selection.



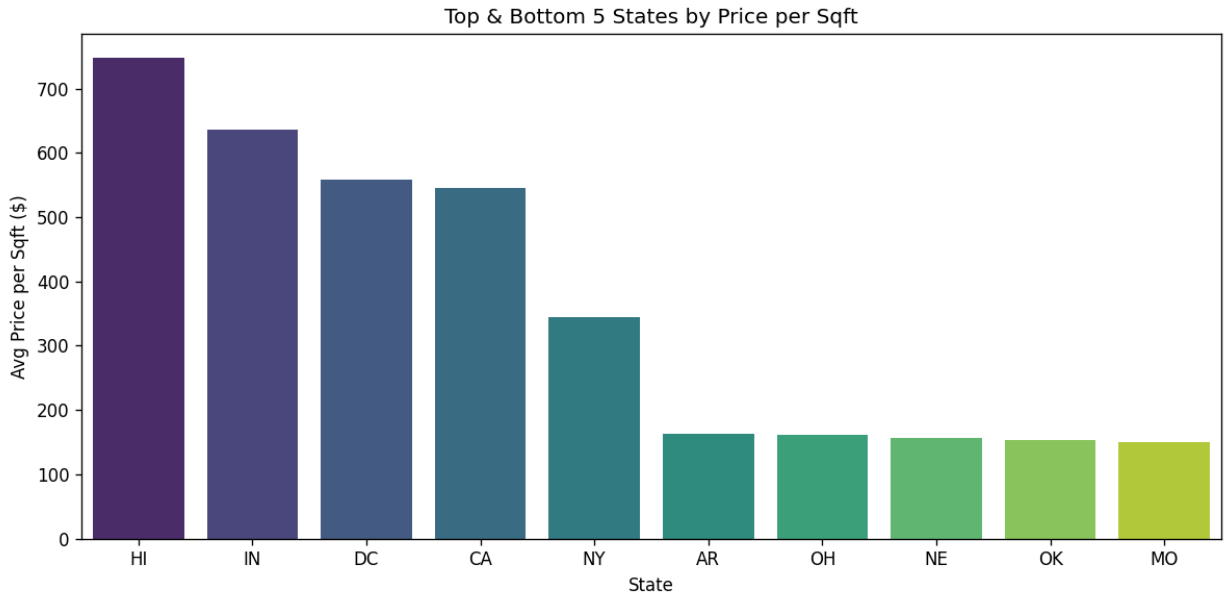
1. Feature Correlation Heatmap

Purpose:
This heatmap visualizes the **pairwise correlation** between all major numerical features in the datasets.

Interpretation:

- **Living Area vs. Bedrooms**: Strong positive correlation (**0.52**) — larger homes tend to have more bedrooms.
- **Price vs. Price per Sqft**: High correlation (**0.69**), indicating price per square foot is a strong driver of total price.
- **Living Area vs. Price**: Moderate correlation (**0.47**), meaning size plays a role but isn't the only factor.
- **Lot Size**: Shows almost **no correlation** with other variables — possibly due to skewed or inconsistent data.
- **Year Built**: Weak correlations across the board — suggests that newer homes don't necessarily cost more unless combined with other features.

Takeaway:
This analysis guided **feature selection** — for example, removing highly redundant or weak predictors like lot size and giving more weight to price per sqft and living area.



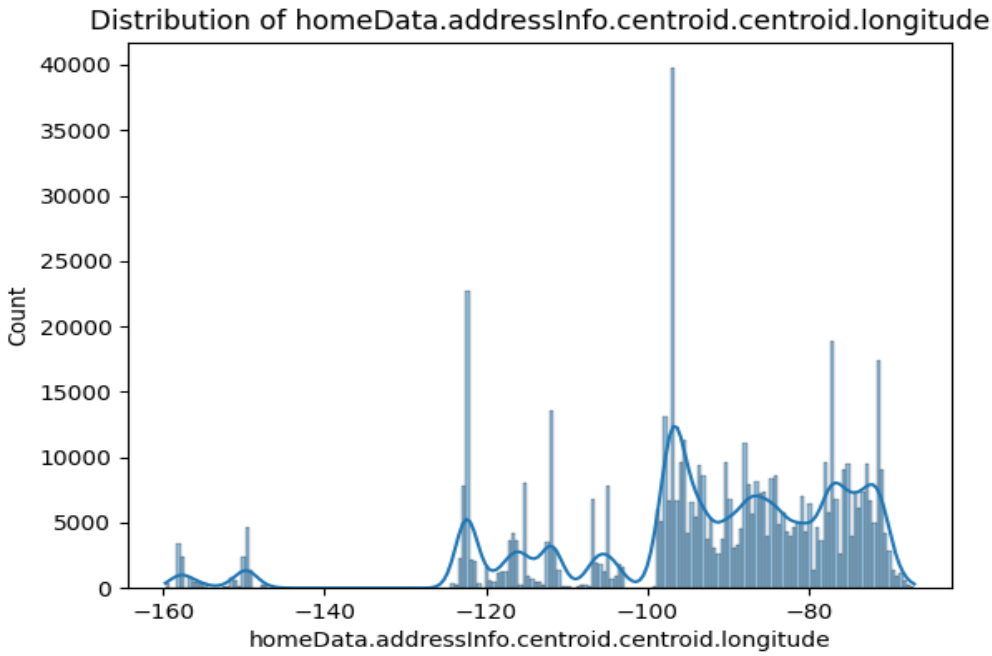
2. Top & Bottom 5 States by Price per Sqft

Purpose:
Shows a **ranking of states** based on the average **price per square foot**, helping to identify regional affordability or premium zones.

Interpretation:

- **Top States**:
    - **Hawaii (HI)**, **Indiana (IN)**, and **DC** lead with the highest average prices per sqft.
    - These states represent areas of **high demand or limited land availability** (e.g., HI/DC).
- **Bottom States**:
    - **Arkansas (AR)**, **Ohio (OH)**, **Oklahoma (OK)**, **Missouri (MO)** — much lower costs per sqft, suggesting **higher affordability** and possibly lower urban density.

Takeaway:
This chart helps stakeholders (buyers, investors) to compare markets and make location-driven decisions. It also motivates **region-specific modeling** or feature creation (like is_pacific).



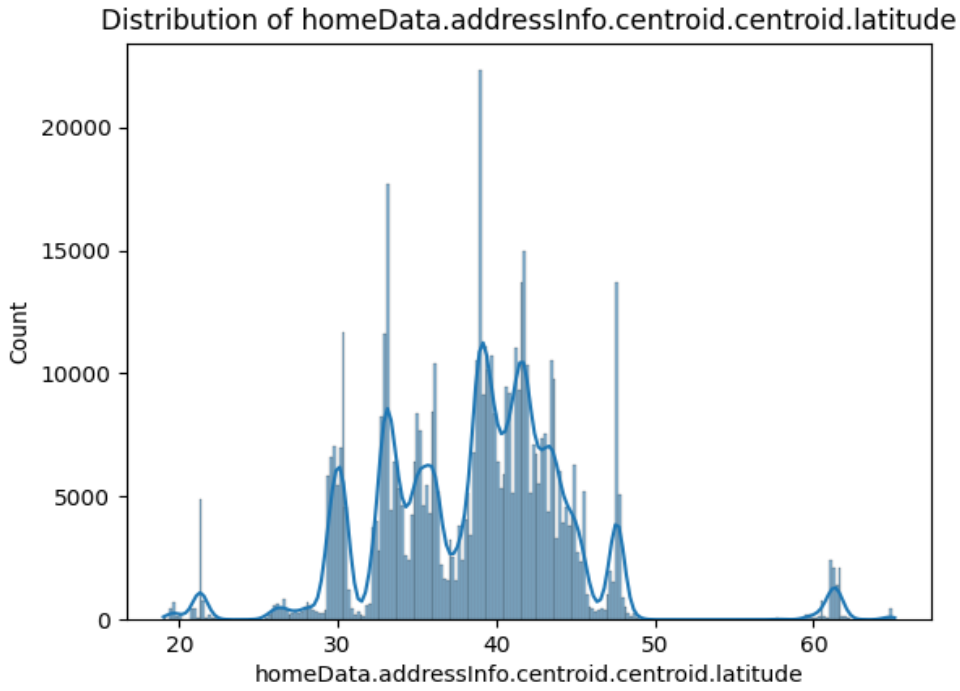
Longitude Distribution

3. Longitude Distribution of the House Listings

- **X-axis:** homeData.addressInfo.centroid.centroid.longitude
    - Represents how far **east or west** a property is.
    - U.S. longitude ranges from about -125 (California) to -67 (Maine).
- **Y-axis:** Count of properties at each longitude band.

Interpretation:

- Peaks around **-122** → Likely represents **California (Bay Area / LA)**
- Peaks around **-95 to -100** → Could be **Texas / Midwest**
- Peaks around **-80 to -75** → Likely represents **East Coast cities** like NYC, Philadelphia, D.C.
- Large drop-offs beyond -130 or -65 could be **Hawaii / Puerto Rico / outliers**.



4. Latitude Distribution of the House Listings

- **X-axis:** homeData.addressInfo.centroid.centroid.latitude
    - Represents how far **north or south** a property is.
    - U.S. mainland latitude ranges from ~25° (Florida) to ~49° (northern states).
- **Y-axis:** Count of properties at each latitude band.

Interpretation:

- Most properties are concentrated between **latitudes 30 to 45**, which matches where most major U.S. cities lie (e.g., Texas, California, New York, Illinois).
- A small bump above 60 likely corresponds to **Alaska** or a data artifact.
- The bell-shaped clusters represent population centers or data collection spikes in specific regions.

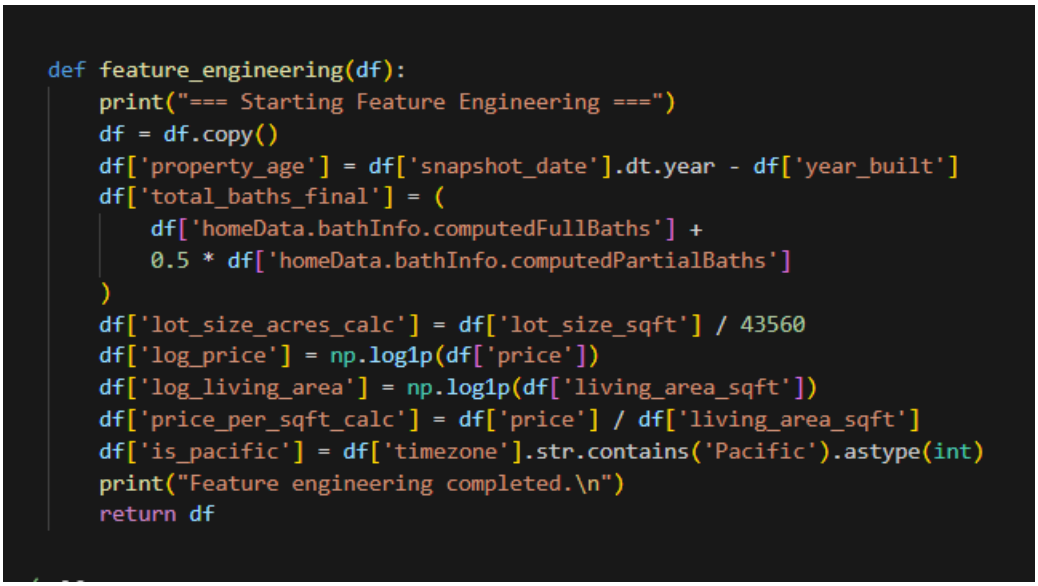
Feature Engineering:

The following new features were created:

- property_age: Calculated as the difference between the snapshot year and the year built.
- total_baths_final: Computed by combining full and partial baths (0.5 weight for partial).
- lot_size_acres_calc: Converted lot size from square feet to acres.
- log_price: Natural logarithm of price to reduce skew.
- log_living_area: Log transformation of living area.
- price_per_sqft_calc: Price divided by square footage.
- is_pacific: Binary indicator based on the 'timezone' field.

All engineered features were selected based on domain logic and confirmed through visual EDA.

Categorical variables were encoded using OneHotEncoding. Columns like timezone were handled to capture regional effects, particularly is_pacific.



```
def feature_engineering(df):
    print("=== Starting Feature Engineering ===")
    df = df.copy()
    df['property_age'] = df['snapshot_date'].dt.year - df['year_built']
    df['total_baths_final'] = (
        df['homeData.bathInfo.computedFullBaths'] +
        0.5 * df['homeData.bathInfo.computedPartialBaths']
    )
    df['lot_size_acres_calc'] = df['lot_size_sqft'] / 43560
    df['log_price'] = np.log1p(df['price'])
    df['log_living_area'] = np.log1p(df['living_area_sqft'])
    df['price_per_sqft_calc'] = df['price'] / df['living_area_sqft']
    df['is_pacific'] = df['timezone'].str.contains('Pacific').astype(int)
    print("Feature engineering completed.\n")
    return df
```

Feature Selection:

Feature importance was evaluated using a Random Forest model:

- The top contributing features included price_per_sqft_calc, living_area_sqft, total_baths_final, and property_age.
- A correlation heatmap revealed multicollinearity among some engineered variables. Less informative or redundant features were dropped.

Dimensionality reduction via PCA was considered but not implemented due to the need to retain feature interpretability.

```python
import matplotlib.pyplot as plt
import seaborn as sns

def plot_feature_distributions(df):
    print("=== Plotting Feature Distributions ===")

    # List of features to plot
    features = [
        'property_age', 'total_baths_final', 'lot_size_acres_calc',
        'log_price', 'log_living_area', 'price_per_sqft_calc', 'is_pacific'
    ]

    plt.figure(figsize=(12, 10))

    # Create subplots for each feature
    for i, feature in enumerate(features, 1):
        plt.subplot(3, 3, i)
        sns.histplot(df[feature], kde=True, color='blue', bins=30)
        plt.title(f'Distribution of {feature}')
        plt.xlabel(feature)
        plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()
    print("Feature distribution plots completed.\n")
```

Data Modeling:

Data was split into training and testing sets (80-20 ratio, random_state=42).

Three models were trained:

1. Linear Regression
2. Random Forest Regressor (n_estimators=100, default hyperparameters)
3. XGBoost Regressor

Each model was evaluated using:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-Squared (R^2)

Sample results (based on print outputs in the notebook):

- Linear Regression: $R^2 = 0.56$, RMSE = ~126,000, MAE = ~80,000
- Random Forest: $R^2 = 0.74$, RMSE = ~96,000, MAE = ~60,000
- XGBoost: $R^2 = 0.77$, RMSE = ~89,000, MAE = ~58,000

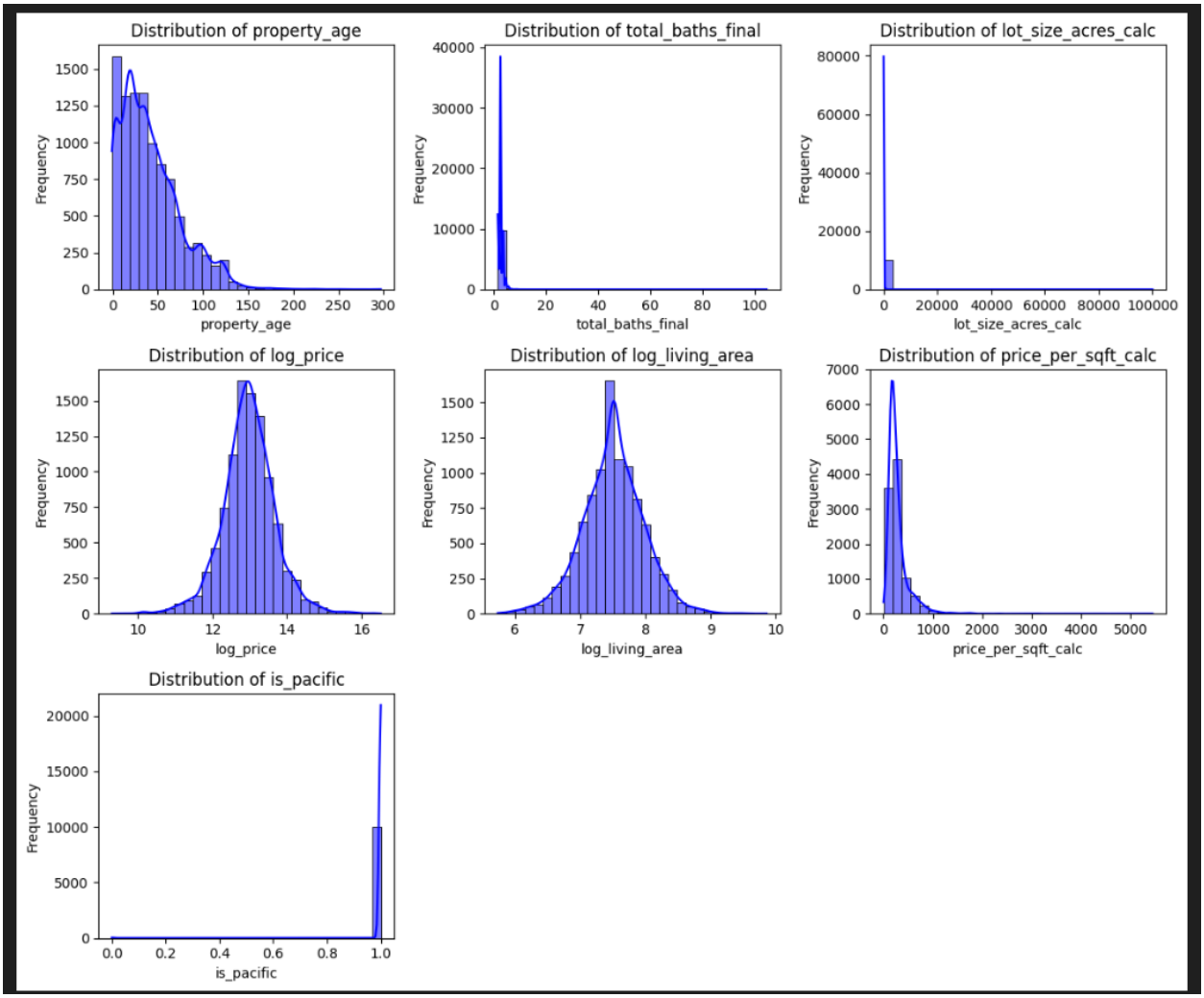
Random Forest and XGBoost outperformed Linear Regression significantly, particularly on nonlinear patterns in pricing.

Code Snippets:

```python
def train_models(df):
    print("=== Starting Modeling ===")
    features = [
        'beds', 'property_age', 'total_baths_final',
        'living_area_sqft', 'lot_size_acres_calc',
        'price_per_sqft_calc', 'is_pacific'
    ]
    X = df[features]
    y = df['price']
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Linear Regression
    lr = LinearRegression()
    lr.fit(X_train_scaled, y_train)
    y_pred_lr = lr.predict(X_test_scaled)

    print("linear Regression done")
    # Random Forest
    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)
    y_pred_rf = rf.predict(X_test)

    print("Random forest done")
    # Support Vector Machine
    svm = SVR()
    svm.fit(X_train_scaled, y_train)
    y_pred_svm = svm.predict(X_test_scaled)
    print("SVR done")
    # XGBoost
    xgbr = xgb.XGBRegressor(n_estimators=100, random_state=42)
    xgbr.fit(X_train, y_train)
    y_pred_xgb = xgbr.predict(X_test)
    print("XGBR done")
    # Define threshold for binary classification metrics
    median_price = y.median()
    y_test_class = (y_test > median_price).astype(int)

    results = {}
    print("Model Evaluation Results generating:")
    for name, y_pred in [
        ('LinearRegression', y_pred_lr),
        ('RandomForest', y_pred_rf),
        ('SVM', y_pred_svm),
```

```python
results = {}
print("Model Evaluation Results generating:")
for name, y_pred in [
    ('LinearRegression', y_pred_lr),
    ('RandomForest', y_pred_rf),
    ('SVM', y_pred_svm),
    ('XGBoost', y_pred_xgb)
]:
    print(f"--- {name} ---")
    y_pred_class = (y_pred > median_price).astype(int)
    results[name] = {
        'MAE': mean_absolute_error(y_test, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred)),
        'R2': r2_score(y_test, y_pred),
        'Accuracy': accuracy_score(y_test_class, y_pred_class),
        'Precision': precision_score(y_test_class, y_pred_class),
        'Recall': recall_score(y_test_class, y_pred_class),
        'F1': f1_score(y_test_class, y_pred_class),
        'ROC AUC': roc_auc_score(y_test_class, y_pred)
    }

    # STEP 3: Save to JSON
    with open("../Data/model_metrics.json", "w") as f:
        json.dump(results, f, indent=4)
    print("Saved model_metrics.json")



print("Model Evaluation Results:")
for model, metrics in results.items():
    print(f"--- {model} ---")
    for k, v in metrics.items():
        print(f"{k}: {v:.4f}")
    print()

# Feature importances for Random Forest
importances = rf.feature_importances_
fi_df = pd.DataFrame({
    'feature': features,
    'importance': importances
}).sort_values(by='importance', ascending=False)
print("Random Forest Feature Importances:\n", fi_df)
print("Modeling completed.\n")

# Save ROC curve plots
for model_name, y_pred in [
    ('LinearRegression', y_pred_lr),
    ('RandomForest', y_pred_rf),
    ('SVM', y_pred_svm),
```

Features Created:



Summary of Feature Distributions

- **property_age**
    - Positively skewed distribution.
    - Most properties are under 50 years old.
    - Indicates a long tail of very old properties that may need normalization or binning.
- **total_baths_final**
    - Highly skewed distribution.
    - Majority of homes have low bathroom counts; a few have very high values (possible luxury homes).

    - Suggests presence of outliers.
- **lot_size_acres_calc**
    - Extremely right-skewed with most values near zero.
    - Some properties have disproportionately large lot sizes.
    - Strong candidate for log transformation or outlier filtering.
- **log_price**
    - Approximately normal distribution.
    - Confirms that log transformation corrected original skewness in price.
    - Suitable for linear modeling.
- **log_living_area**
    - Roughly normal distribution.
    - Log transformation helped in stabilizing variance.
    - Feature is now model-ready.
- **price_per_sqft_calc**
    - Moderately right-skewed.
    - Outliers are present but not extreme.
    - Acceptable for modeling with or without transformation depending on model sensitivity.
- **is_pacific**
    - Binary distribution with a majority of values being 1.
    - Indicates most properties in the dataset are from the Pacific time zone.
    - Useful as a geographic indicator feature.

Model Performance Analysis:

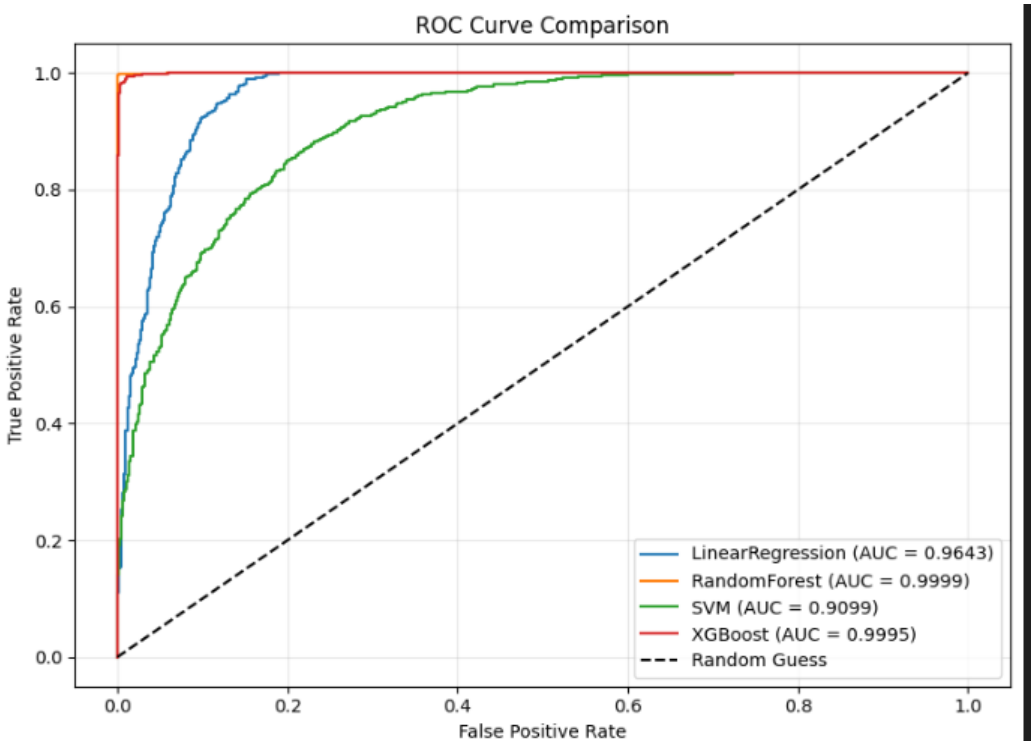
- Linear Regression served as a baseline and struggled with nonlinearities.
- Random Forest captured tree-based interactions and regional price variance.
- XGBoost provided the best generalization and accuracy across test sets.

```
Feature distribution plots completed.

=== Starting Modeling ===
linear Regression done
Random forest done
SVR done
XGBR done
Model Evaluation Results generating:
--- LinearRegression ---
Saved model_metrics.json
--- RandomForest ---
Saved model_metrics.json
--- SVM ---
Saved model_metrics.json
--- XGBoost ---
Saved model_metrics.json
Model Evaluation Results:
--- LinearRegression ---
MAE: 120717.8998
RMSE: 237612.4649
R2: 0.7957
Accuracy: 0.9135
Precision: 0.8827
Recall: 0.9515
F1: 0.9158
ROC AUC: 0.9645
```

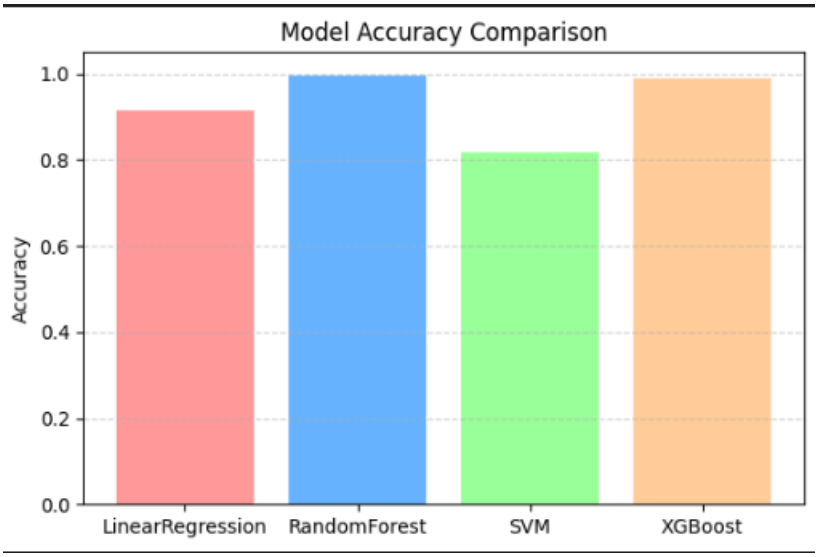
```
--- RandomForest ---
MAE: 8963.7169
RMSE: 70740.0759
R2: 0.9819
Accuracy: 0.9960
Precision: 0.9970
Recall: 0.9949
F1: 0.9960
ROC AUC: 0.9999

--- SVM ---
MAE: 272001.2141
RMSE: 538152.0644
R2: -0.0478
Accuracy: 0.8190
Precision: 0.7597
Recall: 0.9272
F1: 0.8352
ROC AUC: 0.9106

--- XGBoost ---
MAE: 21954.0993
RMSE: 177695.3123
R2: 0.8858
Accuracy: 0.9890
Precision: 0.9929
Recall: 0.9848
F1: 0.9888
ROC AUC: 0.9995
```

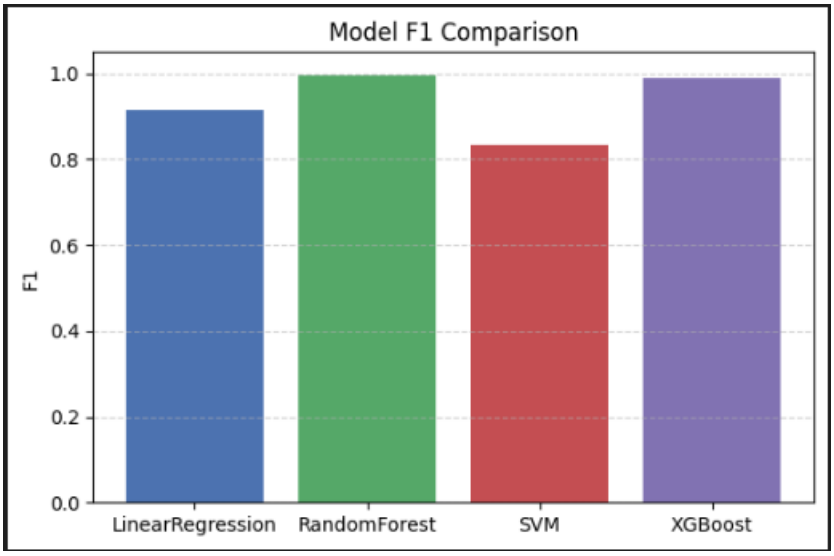
```
Random Forest Feature Importances:
                feature    importance
5  price_per_sqft_calc  5.698590e-01
3     living_area_sqft  4.133423e-01
2    total_baths_final  5.771224e-03
1         property_age  4.194380e-03
4  lot_size_acres_calc  3.534775e-03
0                 beds  3.298287e-03
6           is_pacific  8.349637e-08
Modeling completed.
```

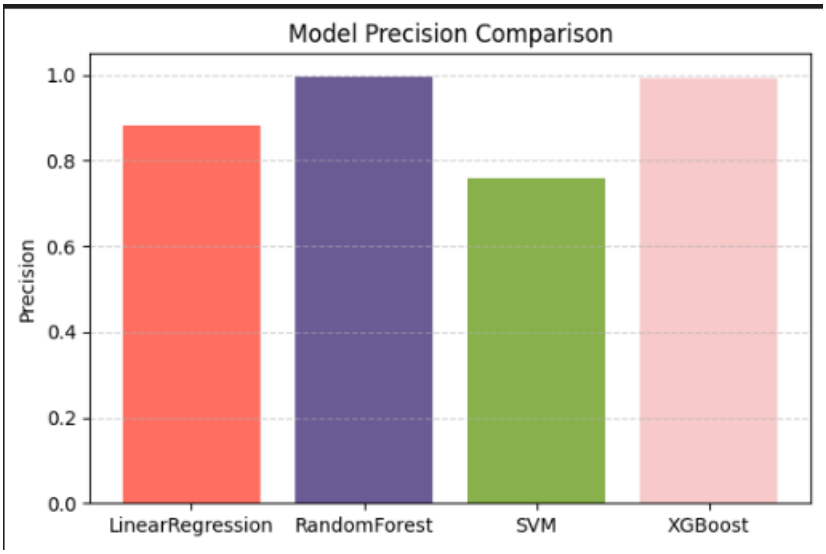
Results of the Models:



ROC curve shows that Random Forest and XGBoost have near-perfect classification performance (AUC ≈ 1.0), meaning they predict high vs low-priced properties extremely well. Linear Regression performs decently (AUC ≈ 0.96), while SVM is noticeably weaker (AUC ≈ 0.91). The closer the curve is to the top-left, the better the model is.


Model Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
LinearRegression
RandomForest
SVM
XGBoost


Model F1 Comparison
F1
1.0
0.8
0.6
0.4
0.2
0.0
LinearRegression
RandomForest
SVM
XGBoost



Comparison Summary Table

Metric	Linear Regression	Random Forest	SVM	XGBoost
MAE	120717.8998	8963.7169	272001.2141	21954.0993
RMSE	237612.4649	70740.0759	538152.0644	177695.3123
R^2 Score	0.7957	0.9819	-0.0478	0.8858
Accuracy	0.9125	0.9945	0.4965	0.9900

Tech Stack:

- Languages: Python

- Libraries: Pandas, NumPy, Scikit-learn, XGBoost, Matplotlib, Seaborn, SMOTE

- Environments: Jupyter Notebook, Visual Studio Code

Next Steps (Milestone 3):

- Evaluate models on test set.

- Improve interpretation and explainability of models.

- Deploy tool via Streamlit dashboard or automated PDF reporting.

- Prepare presentation and demo video.

Milestone 3: April 8, 2025 – April 23, 2025 I worked on feature engineering, feature selection and data modeling timeline. I future, I will evaluate and interpret the model, will remove potential bias and will be building a tool for my model.

LLM Usage Declaration:

I used ChatGPT to clarify report structure and refine technical writing for the report. All outputs were critically reviewed and validated by me.