Capstone Two: Final Project Report

Introduction

Purchasing property is a significant financial decision for most individuals. This analysis focuses on property in California to assist prospective homebuyers, particularly first-time buyers, in evaluating whether purchasing a home would be a beneficial decision.

A grand purchase like a property needs to be completely analyzed and that is what we will do and while be doing it by zeroing in on one property.

Table of Contents

b. df.pivot_table()

3. Splitting the Data

4. Modeling

a. Linear Regression

- param_grid in grid_search

- Lasso

b. ARIMA

- parameter_combinations

- Calculated and compared predictive values to actual values

c. Random Forest

- param_grid in grid_search

- RandomForestRegressor

d. XGBoost

- param_grid in grid_search

- XGBRegressor

5. Comparing the Models

6. Conclusion

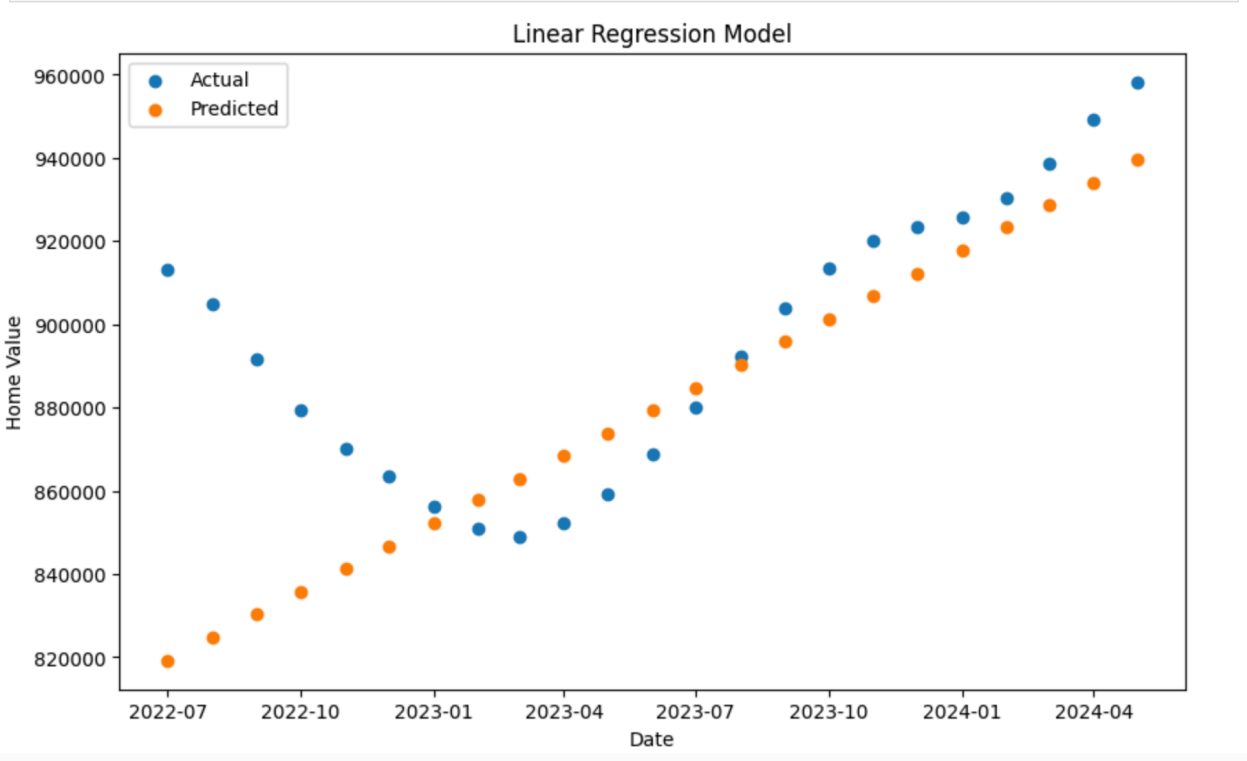
Overview

Thought the process of exploring the data I had to make a few changes and pivot the data to be useful for my future modeling of the data later on. Below, you'll see that I used .melt to take the data frame from a wide format to a long format. This made it easier for me to form the columns to graph it later and organize the dates for times series later on.

```
#Melting Data - melted data allows you to split the dataset by time
df_melt=pd.melt(df, id_vars=['RegionID','SizeRank','Location','State',], var_name='Year_Month_Parameters', 

df_melt[['Year', 'Month_Parameters']] = df_melt['Year_Month_Parameters'].str.split('-', expand=True, n=1)
df_melt.drop('Year_Month_Parameters', axis=1, inplace=True)

#Organize so melted DF shows a column of month
df_melt[['Month', 'Parameters']] = df_melt['Month_Parameters'].str.split('-', expand=True, n=1)
df_melt.drop(columns = ['Month_Parameters'], inplace=True)

df_melt.head()
```

Initially, I explored modeling with Linear Regression and decided to enhance its performance by incorporating hyperparameter tuning. Specifically, I evaluated various regularization techniques, including Ridge Regression, Lasso Regression, and Elastic Net.

After careful consideration, I chose to focus on Lasso Regression for hyperparameter tuning within the Linear Regression framework. However, the results were not promising. The modeling outcomes and associated error rates indicated suboptimal performance, suggesting potential issues with either the approach or the data itself.



In comparison to ARIMA, Random Forest, and XGBoost, Linear Regression emerged as the best modeling approach. I assessed each model's fit by comparing actual values with predicted values, and Linear Regression demonstrated the best overall fit.

Using Linear Regression, we observed a gradual increase in property values in San Diego. This suggests that purchasing property in San Diego would be a sound investment. Over time, homebuyers or investors are unlikely to lose equity and may see property gains.

This analysis is based solely on property values as a univariate time series. It considers only the value of a single home over a six-year period in San Diego and does not incorporate additional variables.

Comparing Modelling

When comparing the modeling, I used the metrics of how well does the modeling fit the data. I used MAE, MSE, RMSE and R Squared to provide insight on how well the data fit each modeling.

- **Mean Absolute Error (MAE)** : Avg of the Abs differences between predictions vs actual values.

- **Mean Squared Error (MSE)**: Avg of the squared difference between predictions vs actual values

- **Root Mean Squared Error (RMSE)** : Squared MSE which represents the error in the same unit

- **R -Squared (R^2)** : Proportions of variance explained by the model which is the better the modeling the close it is to 1.

With this information of going over how each model was compared with this analysis and I will show you the hyper parameter tuning for each model.

Linear Regression / Lasso Regression:

```
param_grid = {
    'alpha': [0.0001, 0.001, 0.01, .1, 1, 10, 100, 1000]
}
lasso_cv = GridSearchCV(lasso, param_grid, cv=5, n_jobs=-1)
lasso_cv.fit(X_train, y_train)
```

```
Lasso Regression CV Mean Absolute Error: 21763.103482213904
Lasso Regression CV Mean Squared Error: 1051165040.4004929
Lasso Regression CV R2 Score: 0.024946047885900935
```

ARIMA:

```
parameter_combinations = [
    (0, 0, 0), (1, 0, 0), (2, 0, 0),
    (0, 1, 0), (1, 1, 0), (2, 1, 0),
    (0, 1, 1), (1, 1, 1), (2, 1, 1),
    (0, 2, 0), (1, 2, 0)
]
```

```
ARIMA Model Metrics:
Mean Absolute Error: 22690.318086534808
Mean Squared Error: 11668804378.281862
Root Mean Squared Error: 108022.24020210774
R² Score: -0.3489949455433512
```

```
Mean Absolute Error: 1255683793.4159882
Mean Squared Error: 29965.081302239945
Root Mean Squared Error: 29965.081302239945
R Squared : -0.16476423617494285
```

Random Forest:

```
param_grid = {
    'n_estimators': [200, 300, 500],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None]
}


grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', verbose=2
grid_search.fit(X_train, y_train)

# Best parameters
print("Best Parameters:", grid_search.best_params_)
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
Best Parameters: {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimato
rs': 200}
```

```
Mean Absolute Error Version 2: 29273.27390214965
Mean Squared Error Version 2: 1157713318.877615
Root Mean Squared Error Version 2: 34025.18653699954
R² Score Version 2: -0.07388745211376557
```

XGBoost:

```
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.1, 0.2]
}

#XGBoost model object
xgb_model = xgb.XGBRegressor(random_state=42)

# Create the GridSearchCV object
grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid_xgb,
    cv=5,
    scoring='neg_mean_squared_error',
    verbose=2,
    n_jobs=-1,
    error_score='raise'
)

#GridSearchCV object
grid_search.fit(X_train, y_train)

#print
print(f"Best set of hyperparameters: {grid_search.best_params_} ")
print(f"Best score (negative MSE): {grid_search.best_score_}")
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
Best set of hyperparameters: {'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 100}
Best score (negative MSE): -2525820905.3814917
```

Future Analysis

To expand this analysis in the future, I would include other variables to assess their impact on property values in San Diego over the same six-year period. For instance, I would analyze how factors such as natural disasters (landslides, floods, and earthquakes) influence property values. Additionally, I would examine the effects of the political climate, including Mello-Roos parameters, COVID-19 restrictions, and permit requirements that affect housing inventory.

This analysis, based solely on home values, serves as the foundation for more comprehensive studies to evaluate the stability and volatility of the San Diego property market under various conditions.

Conclusion

Through this process I was able to reassure that a purchase in California, depending on the location, is an amazing investment. While there's so many variables to look that can affect property but by zeroing on San Diego property I was able to determine that it is a good purchase for the purchase. I was also able to determine that despite me believing the ARIMA fitting the data the best, in actuality, Linear Regression is the best fitting model.