Housing in CA

Should we buy a home in SD?

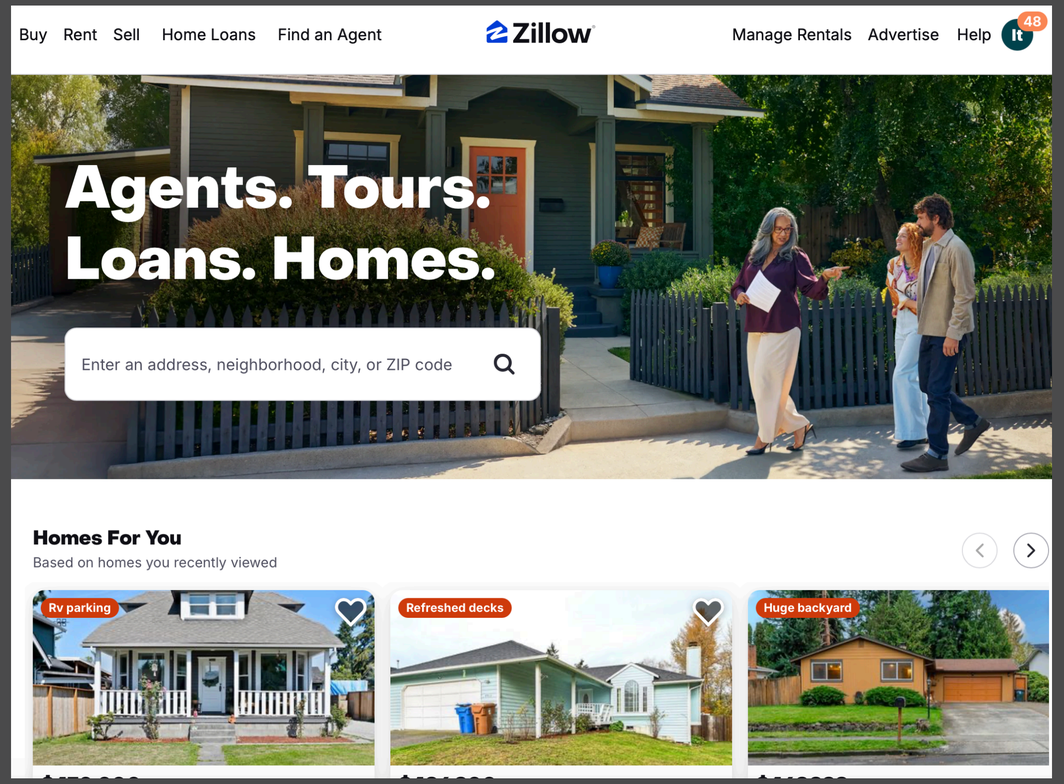
LINDSAY TRUAX

How it started!

- Why I choose this Data?
  - Reassuring that purchasing a home is a good idea
  - Most people's goals are to purchase a home
- Importance of financial freedom
  - Purchasing home will change from
- How does property provide stability?
- Location, Location, Location

Source of the Data

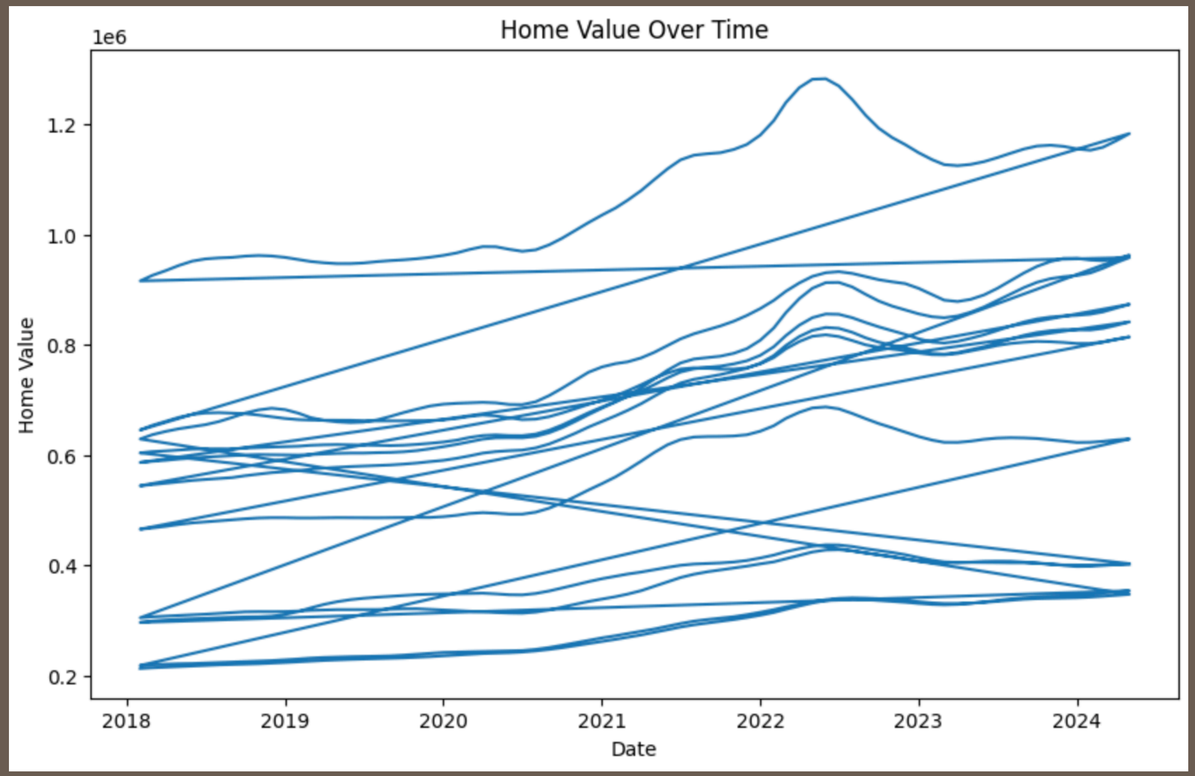
- Reputable site for datasets
  - Kaggle : https://www.kaggle.com/datasets/clovisdalmolinvieira/us-housing-trends-values-time-and-price-cuts
  - US Housing Trends Dataset
  - What is Zillow?



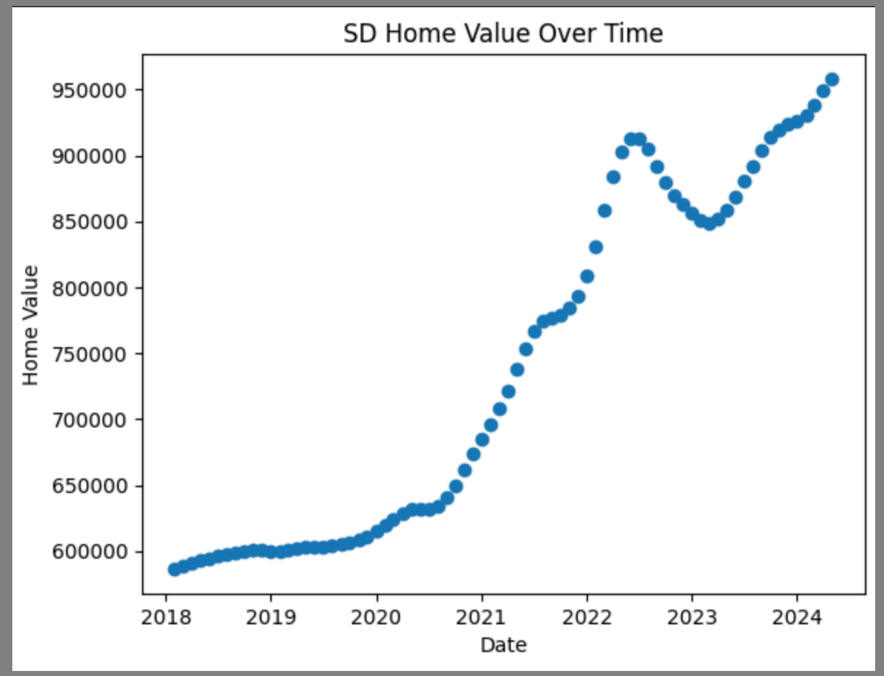
Cleaning the Data

- How did I approach the Data?
  - Renaming columns
  - Dropping values
  - pd.melt()
    - This function in the Pandas library is to reshape the DF
    - Changes the visuals of DF to a Long format
    - Cleans up the dataset
  - Narrowed the view of the data set from US to CA
  - df.unique()
    - Locating the sales per city

Home Values in CA




San Diego, CA



Prepping for Modeling

- Took a look at SD
- What type of Data is this?
  - Univariate Time Series
    - Data point of a single variable (HomeValue) measured overtime
- Splitting the Data

Modeling Done

- Linear Regression
- ARIMA (Autoregressive Integrated Moving Average)
- Random Forest
- XGBoost

How Am I comparing Models?

- MAE
  - AVG magnitude of the error between predicted and actual without direction in mind
- MSE
  - AVG squared difference between the predicted and actual values
- R2
  - How well the model fits the data
    - R^2 = 1 : Perfect
    - R^2 = 0 : No variance
    - R^2 < 0 : Thumbs down

Linear Regression

```
[24] print(f'This is the R2 Score : {clf.score(X_test, y_test)}')

     This is the R2 Score : 0.02507934596610384
```

```
[25] #import to mean squared error
     from sklearn.metrics import mean_squared_error,r2_score
     rmse = np.sqrt(mean_squared_error(y_test, y_pred))
     print(f'This is the root of the Mean Squared Error : {rmse}')

     This is the root of the Mean Squared Error : 32419.459238149804
```

```
y_pred_lasso_cv = lasso_cv.predict(X_test)

mae_lasso_cv = mean_absolute_error(y_test, y_pred_lasso_cv)
print("Lasso Regression CV Mean Absolute Error:", mae_lasso_cv)

mse_lasso_cv = mean_squared_error(y_test, y_pred_lasso_cv)
print("Lasso Regression CV Mean Squared Error:", mse_lasso_cv)

r2_lasso_cv = r2_score(y_test, y_pred_lasso_cv)
print("Lasso Regression CV R2 Score:", r2_lasso_cv)

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

$$y = \beta_0 + \beta_1 x + \varepsilon$$

- Minimize the difference between Predictive values VS Actual values
- Models Relationship between Predictive value VS Actual values
- The Goal is to find the hyperplane in higher dimensions that best fits the data

ARIMA

```python
# Calculate error metrics
mae_ARIMA = mean_absolute_error(y_train, y_pred)
mse_ARIMA = mean_squared_error(y_train, y_pred)
rmse_ARIMA = np.sqrt(mse_ARIMA)
r2_ARIMA = r2_score(y_train, y_pred)

print("ARIMA Model Metrics:")
print("Mean Absolute Error:", mae_ARIMA)
print("Mean Squared Error:", mse_ARIMA)
print("Root Mean Squared Error:", rmse_ARIMA)
print("R² Score:", r2_ARIMA)


#plot
fitted_model.plot_diagnostics(figsize=(12, 8))
plt.show()
```

```
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2      7.652e+06   1.07e+06      7.171      0.000    5.56e+06    9.74e+06
===================================================================================
Ljung-Box (L1) (Q):                  23.40   Jarque-Bera (JB):                 9.36
Prob(Q):                              0.00   Prob(JB):                         0.01
Heteroskedasticity (H):              72.43   Skew:                            -0.39
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.95
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step
ARIMA Model Metrics:
Mean Absolute Error: 22690.318086534808
Mean Squared Error: 11668804378.281862
Root Mean Squared Error: 108022.24020210774
R² Score: -0.3489949455433512
```

$$Y_t - 2Y_{t-1} + Y_{t-2} = \varepsilon_t$$

- AutoRegressive Integrated Moving Average
- Used for time series and forecasting
- ARIMA Model : (0,2,0)
  - Stabilized variance
  - Makes the series Stationary

Random Forest

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i$$

```
# Calculate regression metrics
mae = mean_absolute_error(y_test, y_pred_rf)
mse = mean_squared_error(y_test, y_pred_rf)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred_rf)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R² Score:", r2)
```

```
Mean Absolute Error: 29268.977738766785
Mean Squared Error: 1157126476.148256
Root Mean Squared Error: 34016.561791989734
R² Score: -0.0733431005604519
```

- Uses Bootstrap Sampling
  - Algorithm generates multiple subsets of the training data
  - Each subset is used to train 1 decision tree
- Random Feature : Random Subset of features
- Building Decision Tree : each tree grows to max depth
- Aggregation : Regression, AVG of all Tree depths
- Pro: Combines predictions of multiple trees
- Con: Risks over fitting

XGBoost

$$\text{Objective} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

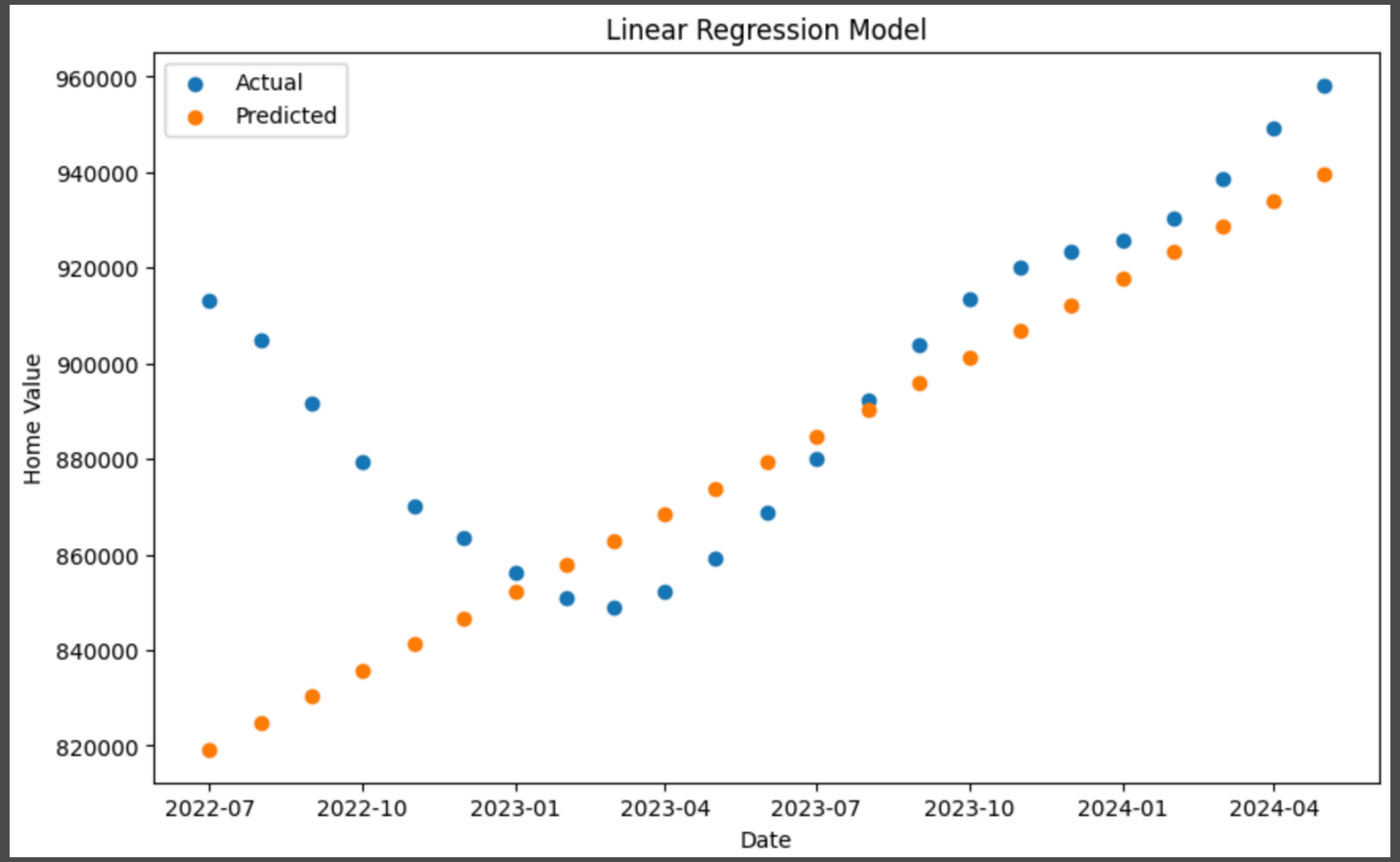
- Extreme Gradient Boosting
- Builds an ensemble of decision trees by adding trees to correct the errors of the previous one
- Pro: Handles Large Datasets
- Con: requires carful hyper parameter tuning
- Objective
  - Loss function
  - Regularization

```
#calculate the regression
mae_xgb = mean_absolute_error(y_test, predictions_xgb)
mse_xgb = mean_squared_error(y_test, predictions_xgb)
rmse_xgb = np.sqrt(mse)
r2_xgb = r2_score(y_test, predictions_xgb)


print(f'Mean Absolute Error: {mse_xgb}')
print(f'Mean Squared Error: {mae_xgb}')
print(f'Root Mean Squared Error: {mae_xgb}')
print(f'R Squared : {r2_xgb}')
```

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

My favorite


Linear Regression Model
Actual
Predicted
960000
940000
920000
900000
880000
860000
840000
820000
Home Value
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
2024-04
Date

Something to take consideration

- Shows the progression of the data over time
  - Time series

- This analysis doesn’t take into account external variables that might affect HomeValue

Main Questions

- Should I have boughten a home in San Diego CA?
- Will my home hold it’s value?

Conclusion

YES!