REAL ESTATE BUYER'S OFFER TOOL BOT USING MACHINE LEARNING

SCENARIO

- We are approached by our same client. She has decided that instead of investing in stocks, she wants to invest in the housing market in Seattle.
- She wants to know what would be a competitive offer for buying a desirable piece of real estate.
- We created a tool to assist our client in making a competitive offer

USER STORY AND ACCEPTANCE CRITERIA

- As a real estate investor, I want to find a way to make a competitive offer based on historical data that takes into consideration site/dwelling features so that my offer is the most desirable to the seller.
- Given the analysis of the Seattle market, we will create a bot that will recommend an offer amount based on the best machine learning model and how aggressive the client wants to make their offer.

METHOD

- We will analyze 5 years of historical data from May 2017 to May 2022, in Seattle, using three different types of machine learning models: Decision Tree, Random Forest, and Linear Regression.
- Specific features include Zip Code, Bathrooms, Bedrooms, Lot Square Footage, Listing Price, Sold to List Price Percentage, Square Footage, and Property Type
- We will then use the analysis to determine the best machine learning model to use in our lex bot
- The bot will then suggest an offer price that our client would offer for a property based on our model and user input.

TECHNOLOGIES
- IMPORTS

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots

from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, OrdinalEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import scipy.stats
```

TECHNOLOGIES
- IMPORTS FOR LAMBDA FUNCTION

```
import pickle
import boto3
import pandas as pd
from boto3.session import Session
from sklearn.tree import DecisionTreeRegressor

from datetime import datetime
from dateutil.relativedelta import relativedelta



import logging
```

TECHNOLOGIES
- AWS

- Lambda
- S3
- Lex Bot
- Cloud Watch

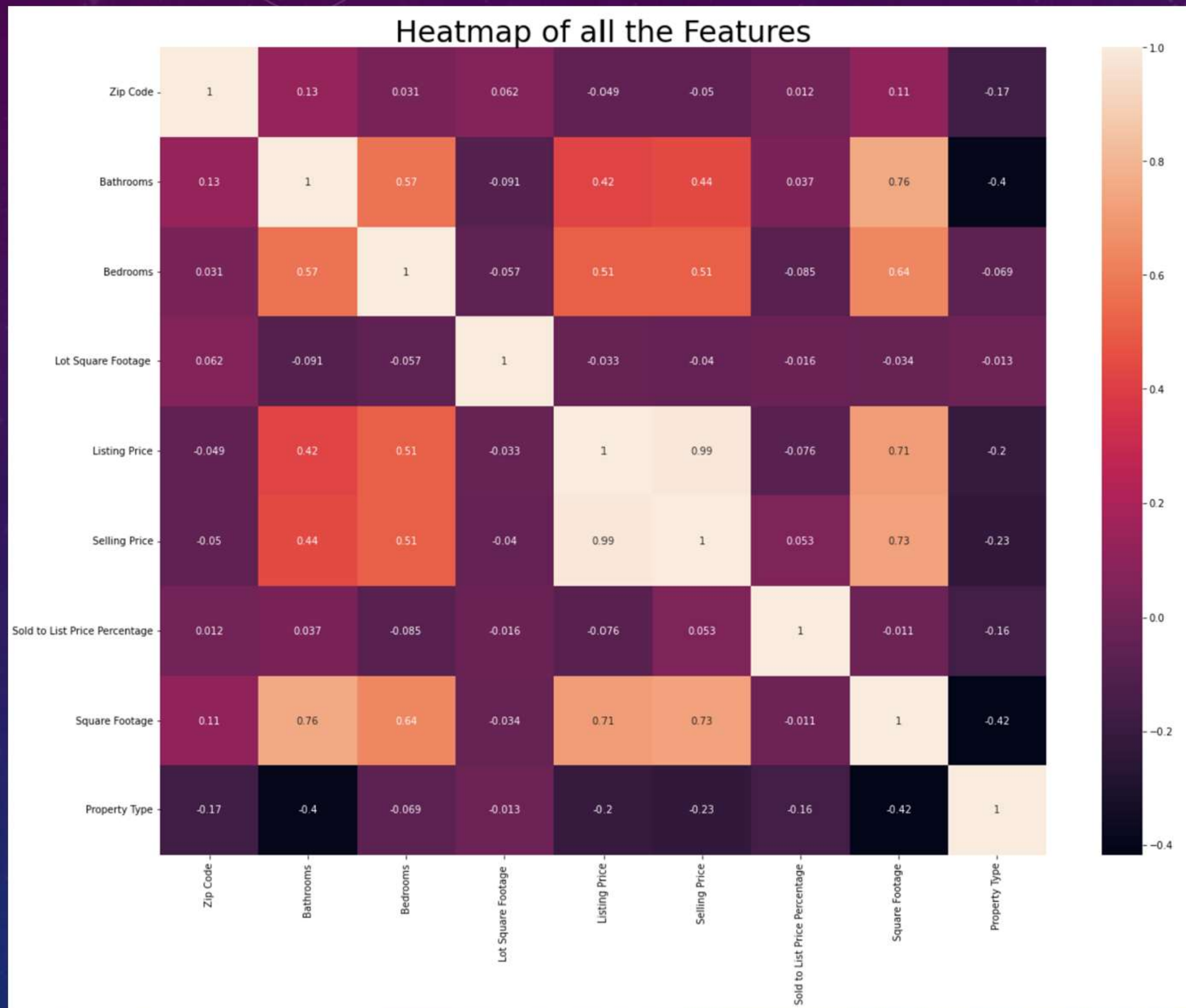
DATA FRAME (RAW)

```
data = pd.read_csv('Sold_And_Stats_Edited_New_a.csv')
data.head()
```

	Listing Number	Street Number	Street Number Modifier	Street Direction	Street Name	Street Suffix	Street Post Direction	City	State	Zip Code	...	Bathrooms	Bedrooms	Lot Square Footage	Listing Price	Selling Price	Sold to List Price Percentage	Square Footage	Style Code	Property Type	Selling Date
0	825199	1118	NaN	NaN	Alki	Ave	SW	Seattle	WA	98116	...	0.0	3	4080.0	1900000.0	1500000.0	78.947368	1000	10 - 1 Story	House	5/1/2017 0:00
1	902993	12805	NaN	NaN	78th	Ave	S	Seattle	WA	98178	...	0.0	0	10500.0	159000.0	155000.0	97.484277	580	10 - 1 Story	House	3/21/2018 0:00
2	1072254	810	NaN	NaN	34th	Ave	NaN	Seattle	WA	98122	...	0.0	0	4600.0	650000.0	650000.0	100.000000	1060	10 - 1 Story	House	5/5/2017 0:00
3	1106354	8735	NaN	NaN	1st	Ave	NW	Seattle	WA	98117	...	0.0	0	6350.0	410000.0	448000.0	109.268293	870	10 - 1 Story	House	5/16/2017 0:00
4	1110111	10403	NaN	NaN	15th	Ave	NaN	Seattle	WA	98125	...	0.0	0	6120.0	498000.0	475000.0	95.381526	1550	10 - 1 Story	House	5/15/2017 0:00

DATA FRAME (CLEANED)

	Zip Code	Bathrooms	Bedrooms	Lot Square Footage	Listing Price	Selling Price	Sold to List Price Percentage	Square Footage	Property Type
132	98126	2	1	6120.0	349950	415000	118.588370	890	1
133	98102	1	1	12269.0	1800000	1800000	100.000000	644	1
134	98136	2	1	4383.0	525000	525000	100.000000	720	1
135	98177	1	1	3333.0	449000	429500	95.657016	550	1
136	98118	3	1	7920.0	299950	299950	100.000000	1320	1


Heatmap of all the Features
Zip Code
Bathrooms
Bedrooms
Lot Square Footage
Listing Price
Selling Price
Sold to List Price Percentage
Square Footage
Property Type
1 0.13 0.031 0.062 -0.049 -0.05 0.012 0.11 -0.17
0.13 1 0.57 -0.091 0.42 0.44 0.037 0.76 -0.4
0.031 0.57 1 -0.057 0.51 0.51 -0.085 0.64 -0.069
0.062 -0.091 -0.057 1 -0.033 -0.04 -0.016 -0.034 -0.013
-0.049 0.42 0.51 -0.033 1 0.99 -0.076 0.71 -0.2
-0.05 0.44 0.51 -0.04 0.99 1 0.053 0.73 -0.23
0.012 0.037 -0.085 -0.016 -0.076 0.053 1 -0.011 -0.16
0.11 0.76 0.64 -0.034 0.71 0.73 -0.011 1 -0.42
-0.17 -0.4 -0.069 -0.013 -0.2 -0.23 -0.16 -0.42 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

CODE FOR MODELS

```
##### LINEAR REGRESSION MODEL #####

reg = LinearRegression()
reg.fit(X_train, y_train)

print('coef of determination training ', reg.score(X_train, y_train))
print('coef of determination testing ', reg.score(X_test, y_test))
print()
reg_pred = list(reg.predict(X_test))
for i in reg_pred[0:10]:
  print('Prediction price of house', reg_pred.index(i)+1, ': $', i)
print()
for i in list(y_test[0:10]):
  print('Real price of house', list(y_test).index(i)+1, ': $', i)
print()
reg_mae = reg.predict(X_train)
print('Mean Absolute Error: ',mean_absolute_error(y_train, reg_mae))
mse = mean_squared_error(y_test,reg_pred)
print('Root Mean Square Error : ', np.sqrt(mse))
```

```
##### DECISION TREE MODEL #####

dt = DecisionTreeRegressor(max_depth=18)
dt.fit(X_train, y_train)

print('coef of determination training ',dt.score(X_train, y_train))
print('coef of determination testing ',dt.score(X_test, y_test))
print()
print('prediction')
dt_pred = list(dt.predict(X_test))
for i in dt_pred[0:10]:
  print('Prediction price of house', dt_pred.index(i)+1, ': $', i)
print()
for i in list(y_test[0:10]):
  print('Real price of house', list(y_test).index(i)+1, ': $', i)
print()
dt_mae = dt.predict(X_train)
print('Mean Absolute Error: ', mean_absolute_error(y_train, dt_mae))
mse = mean_squared_error(y_test,dt_pred)
print('Root Mean Square Error : ', np.sqrt(mse))
```

```
##### RANDOM FOREST MODEL #####

rf = RandomForestRegressor()
rf.fit(X_train, y_train)

print('coef of determination training ',rf.score(X_train, y_train))
print('coef of determination testing ',rf.score(X_test, y_test))
print()
print('prediction')
rf_pred = list(rf.predict(X_test))
for i in rf_pred[0:10]:
  print('Prediction price of house', rf_pred.index(i)+1, ': $', i)
print()
for i in list(y_test[0:10]):
  print('Real price of house', list(y_test).index(i)+1, ': $', i)
print()
rf_mae = rf.predict(X_train)
print('Mean Absolute Error: ', mean_absolute_error(y_train, rf_mae))

mse = mean_squared_error(y_test,rf_pred)
print('Root Mean Square Error : ', np.sqrt(mse))
```

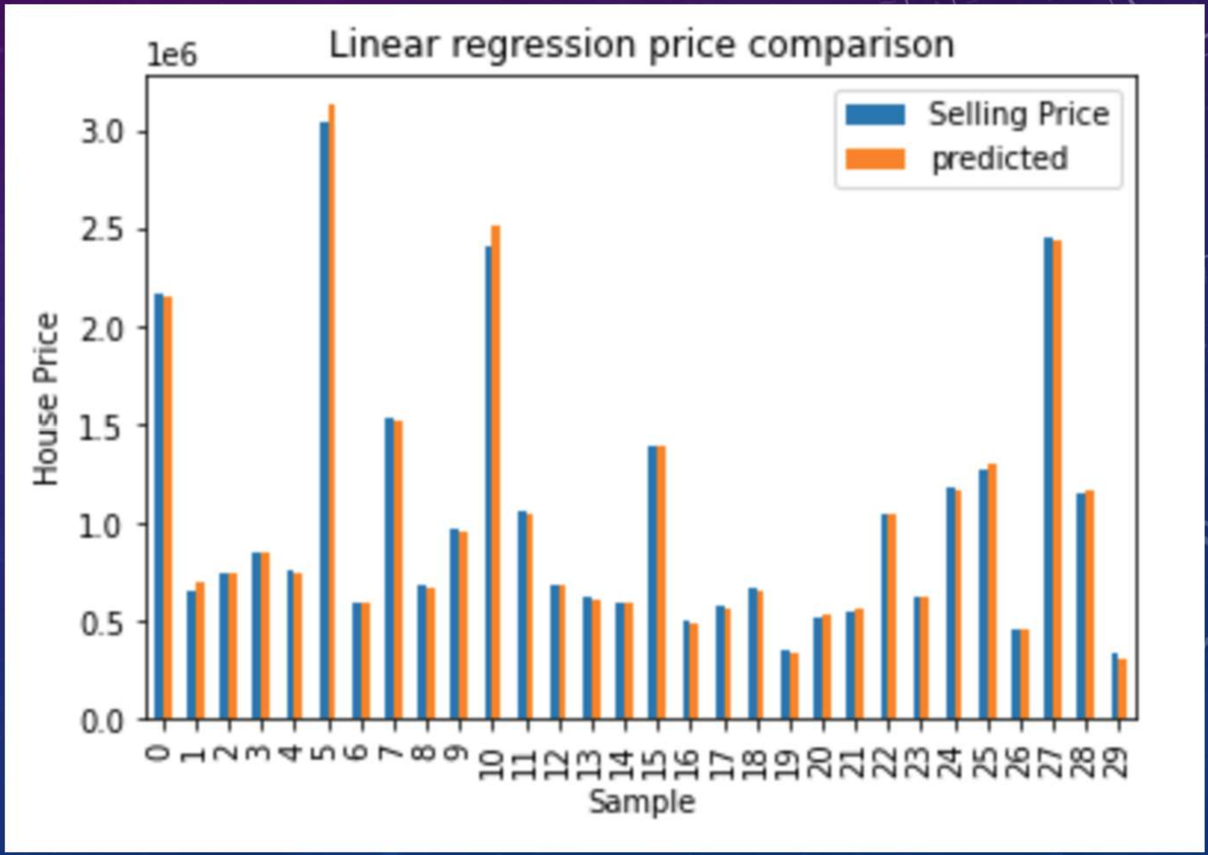
LINEAR REGRESSION MODEL RESULTS

```
coef of determination training  0.9930842826774566
coef of determination testing  0.9855108079557873

Prediction price of house 1 : $ 2152211.825971588
Prediction price of house 2 : $ 691567.4964561757
Prediction price of house 3 : $ 744703.9693826361
Prediction price of house 4 : $ 851219.914802879
Prediction price of house 5 : $ 741933.9407319283
Prediction price of house 6 : $ 3134643.486891551
Prediction price of house 7 : $ 585674.6604139482
Prediction price of house 8 : $ 1517497.969299925
Prediction price of house 9 : $ 663120.2615905219
Prediction price of house 10 : $ 960761.054850621

Real price of house 1 : $ 2168000
Real price of house 2 : $ 650000
Real price of house 3 : $ 750000
Real price of house 4 : $ 850000
Real price of house 5 : $ 757000
Real price of house 6 : $ 3050000
Real price of house 7 : $ 585000
Real price of house 8 : $ 1525000
Real price of house 9 : $ 680000
Real price of house 10 : $ 965000

Mean Absolute Error:  18840.500041941756
Root Mean Square Error :  71854.37970549292
```



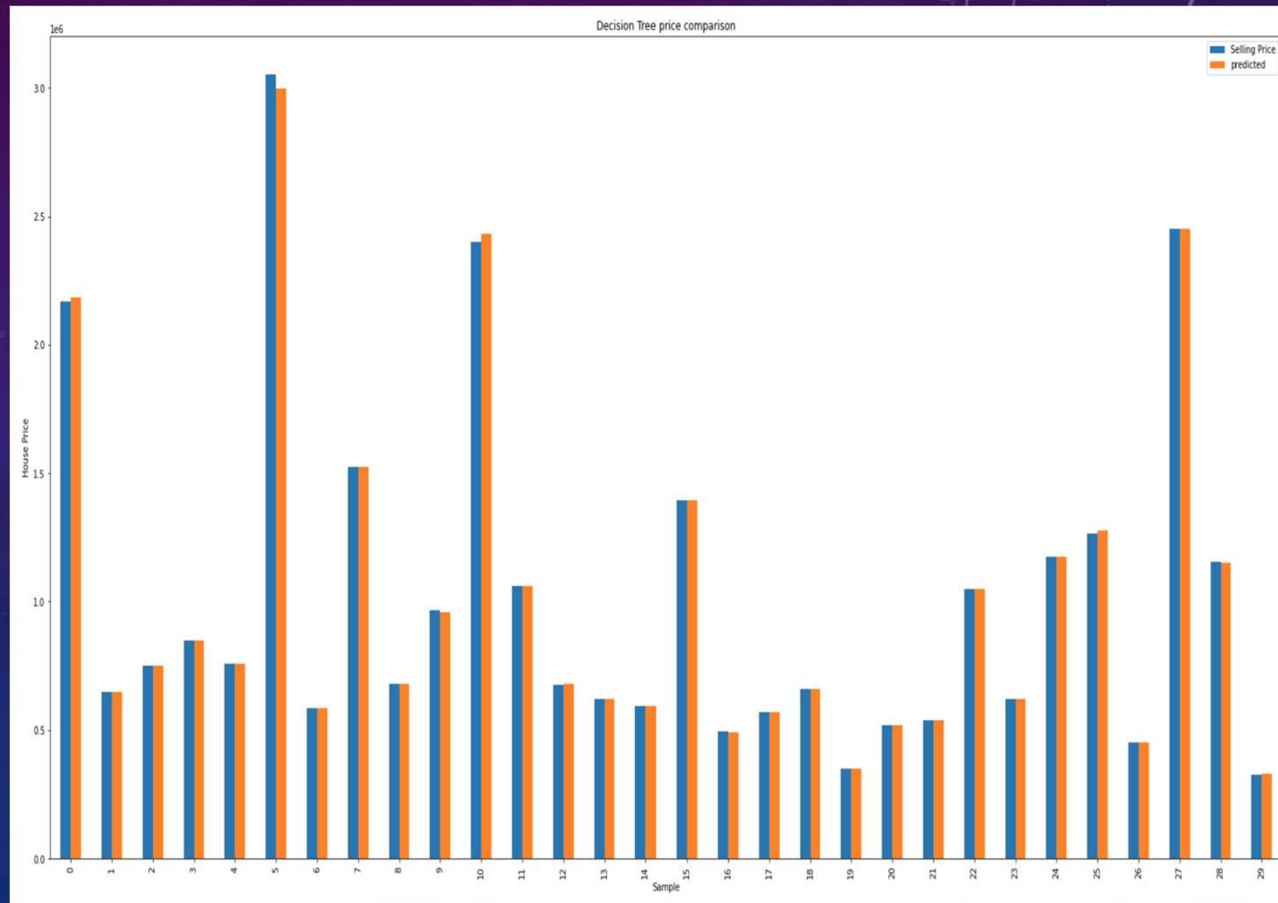
DECISION TREE MODEL RESULTS

```
coef of determination training  0.9999998909838653
coef of determination testing  0.9430661593569819

prediction
Prediction price of house 1 : $ 2185350.0
Prediction price of house 2 : $ 650000.0
Prediction price of house 3 : $ 750000.0
Prediction price of house 4 : $ 849972.7469879518
Prediction price of house 5 : $ 757000.0
Prediction price of house 6 : $ 2995000.0
Prediction price of house 7 : $ 584991.0256410256
Prediction price of house 8 : $ 1525000.0
Prediction price of house 9 : $ 680000.0
Prediction price of house 10 : $ 960000.0

Real price of house 1 : $ 2168000
Real price of house 2 : $ 650000
Real price of house 3 : $ 750000
Real price of house 4 : $ 850000
Real price of house 5 : $ 757000
Real price of house 6 : $ 3050000
Real price of house 7 : $ 585000
Real price of house 8 : $ 1525000
Real price of house 9 : $ 680000
Real price of house 10 : $ 965000

Mean Absolute Error:  32.987184993646956
Root Mean Square Error :  142434.89539722865
```



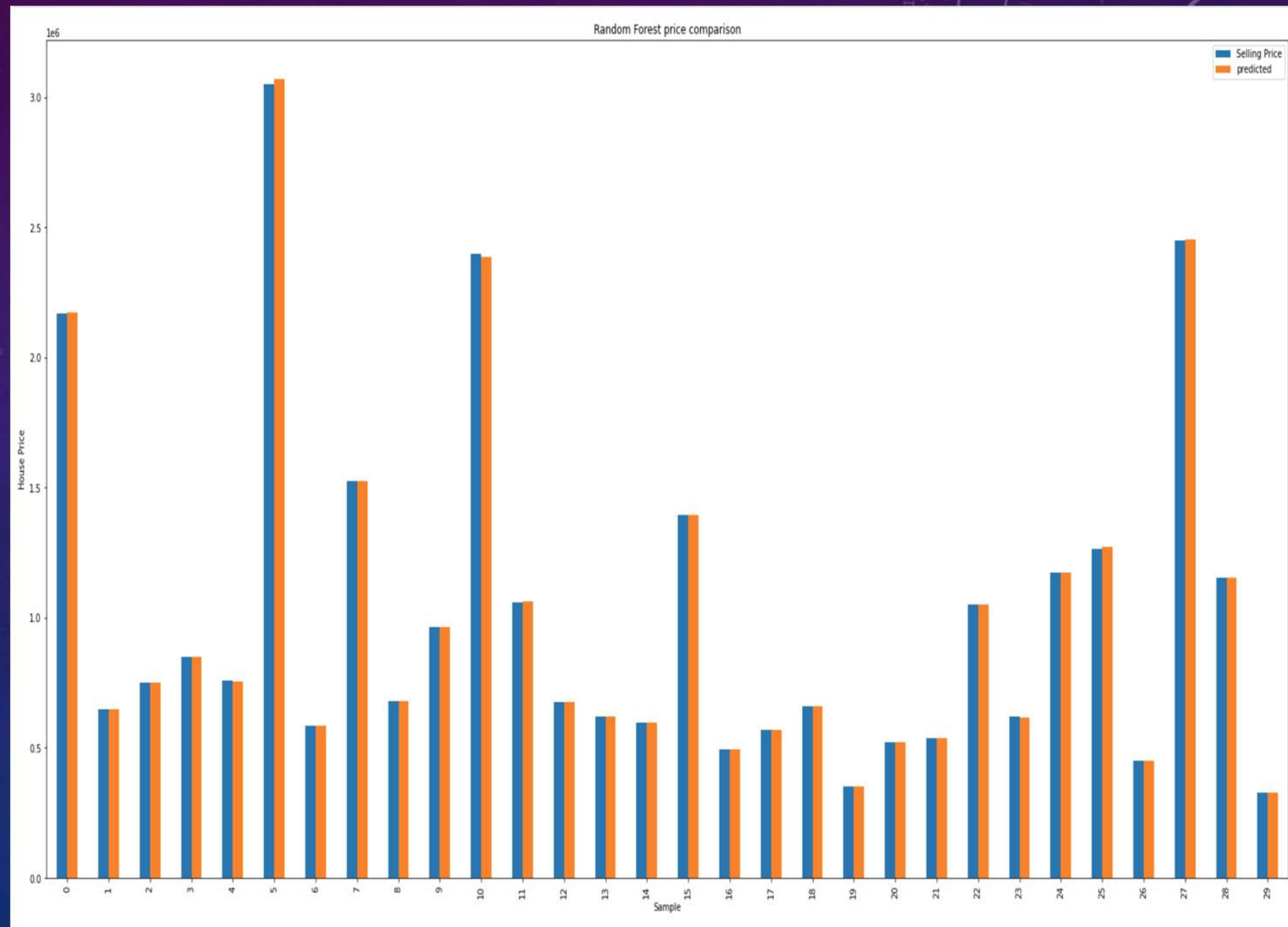
RANDOM FOREST MODEL RESULTS

```
coef of determination training  0.9995700696905242
coef of determination testing  0.9402809175418517

prediction
Prediction price of house 1 : $ 2174259.0
Prediction price of house 2 : $ 649997.12
Prediction price of house 3 : $ 750034.5
Prediction price of house 4 : $ 850000.0
Prediction price of house 5 : $ 756559.72
Prediction price of house 6 : $ 3068152.0
Prediction price of house 7 : $ 585000.0
Prediction price of house 8 : $ 1525380.0
Prediction price of house 9 : $ 680217.38
Prediction price of house 10 : $ 965565.0

Real price of house 1 : $ 2168000
Real price of house 2 : $ 650000
Real price of house 3 : $ 750000
Real price of house 4 : $ 850000
Real price of house 5 : $ 757000
Real price of house 6 : $ 3050000
Real price of house 7 : $ 585000
Real price of house 8 : $ 1525000
Real price of house 9 : $ 680000
Real price of house 10 : $ 965000

Mean Absolute Error:  963.3113261843708
Root Mean Square Error :  145877.30235212093
```

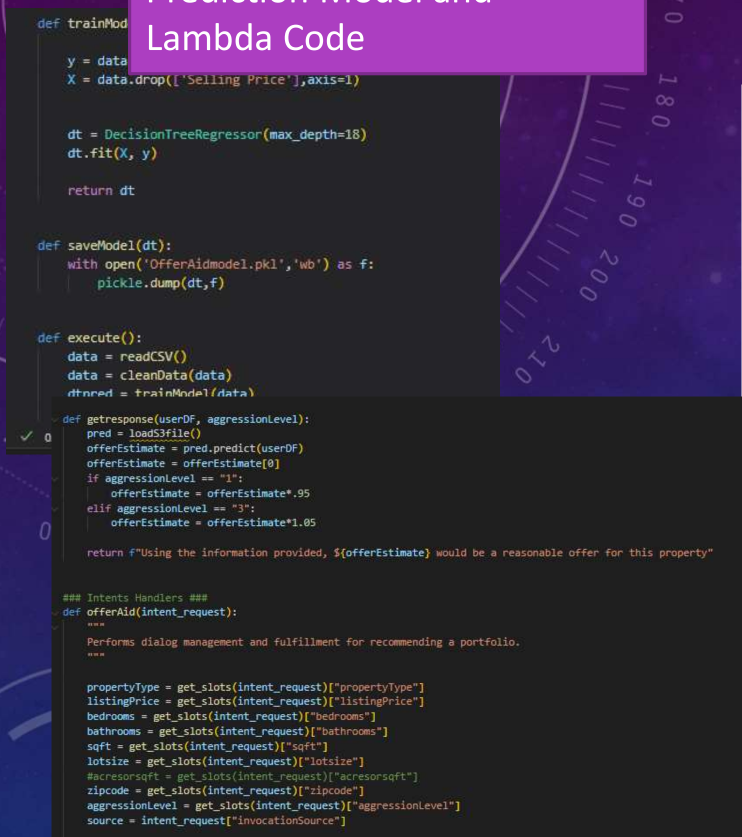


ANALYSIS

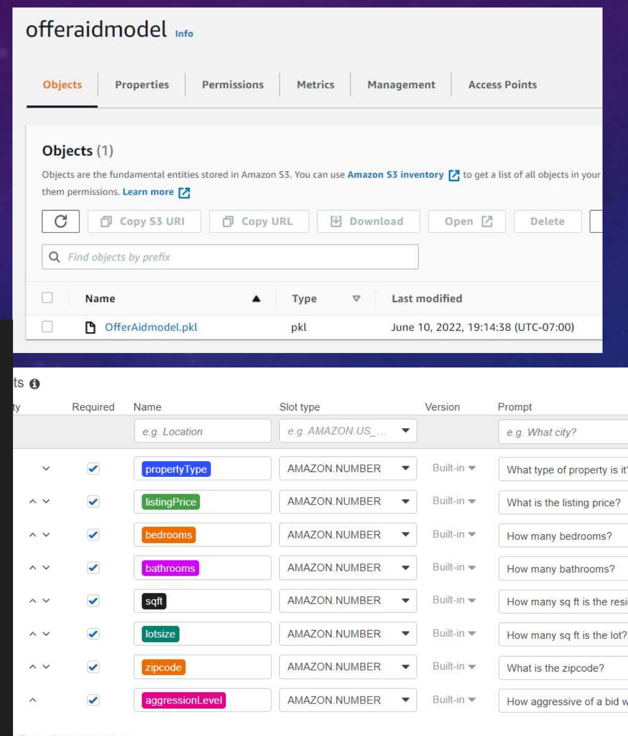
- Based on the Mean Absolute Error, we chose to use the Decision Tree Model with our lex bot
- Decision trees support non linearity, where LR supports only linear solutions. When there are a large number of features with fewer data-sets (with low noise), linear regressions may outperform Decision trees/random forests. In general cases, Decision trees will have better average accuracy.

Structuring and Deploying the Code on AWS

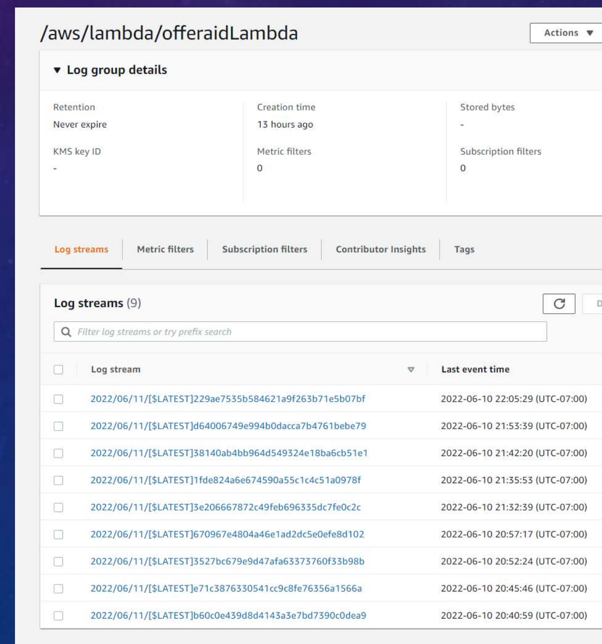
Step 1: Building the Prediction Model and Lambda Code



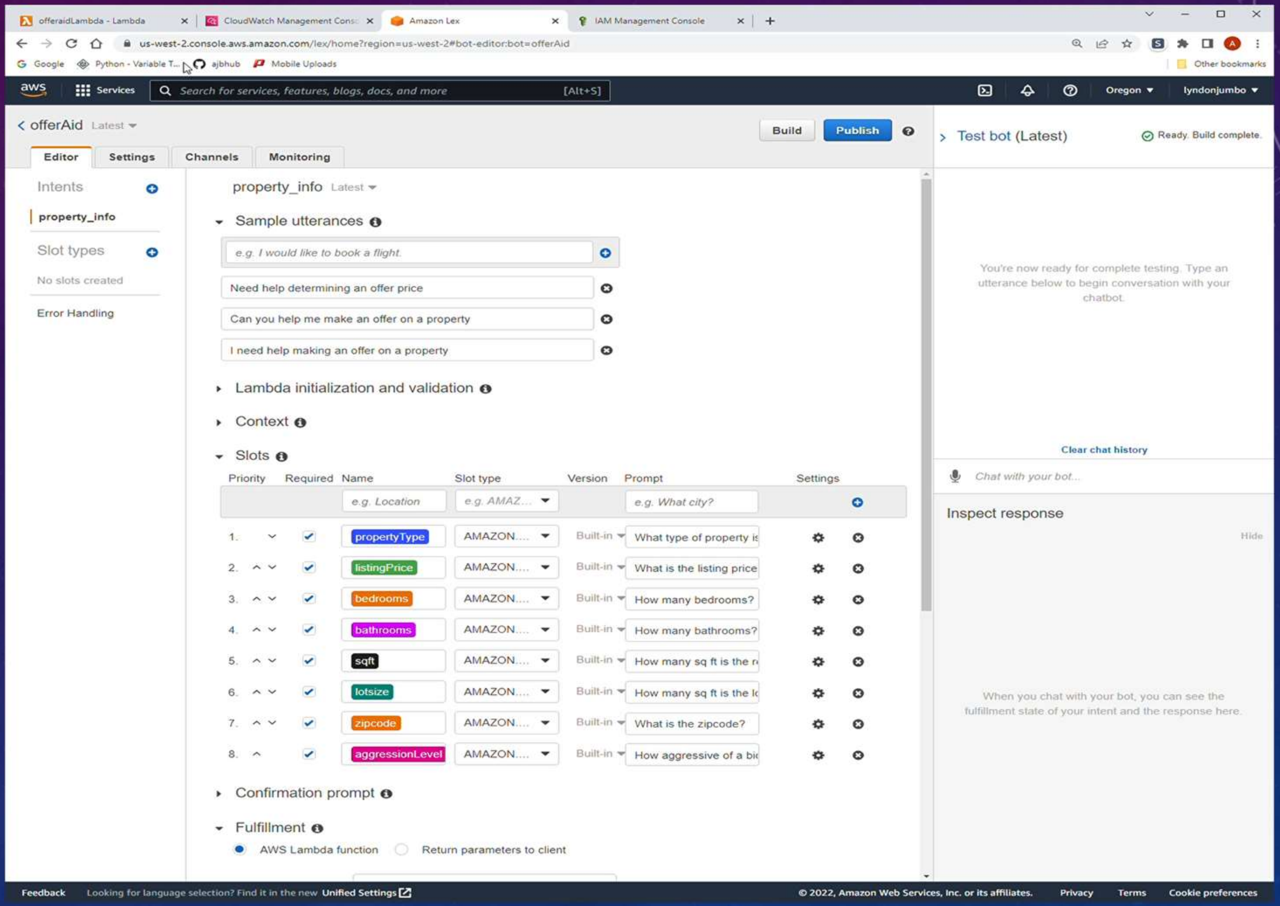
Step 2: Storing trained Prediction model in AWS, Building Lex intent



Step 3: Troubleshooting Compatibility Issues, Permissions, Lambda Layers, compromised access keys



FULL APPLICATION WITH BOT



CONCLUSION

- The bot we have created enables users to, after having supplied desired features, give accurate offers for real estate in the Seattle area.

LIMITATIONS AND NEXT STEPS

- Real estate is an everchanging market that can be hard to predict, with prices that can be based as much on feel as on clear data points.
- There simply is not enough data for an algorithm to learn about longer real estate busts and booms.
- The prediction model doesn't account for emotional/sentimental values. For example, some people will pay more money to purchase a childhood home or a home close to family.
- We could expand the area outside Seattle, but in order to do so, we would need more data. Until that data is procured, the model accuracy only works for Seattle zip codes

CONTRIBUTORS

- Cody Schroeder, codeman@uw.edu
- Hilary Willis, hilarywillis@gmail.com
- Theo Prentice, theoprentice14@gmail.com
- Aaron Bumgarner, aaron.j.bumgarner@gmail.com
- Aranda Furth, arandafurth@gmail.com