Predicting Foreclosure Rates in Los Angeles

Presented by Cesar Ayala, Erik Blood, Han Linwu, Jivan Karapetian, Wang Xiaohang

The California Housing Crisis: An Introduction

- The California Housing Crisis has affected residents of California since 1970, spanning over 5 decades at this point.
- This crisis stems from California's inability to provide an adequate number of homes relative to California's increasing population, leading to increasing housing prices.
- As a domino effect, increasing housing prices led to higher foreclosure rates in the state of California, rising 115.7% from 2021 to 2022.
- While Los Angeles is our focus for this model, it is important to illustrate the genesis of California's Housing Crisis since Los Angeles is a significant part of the Golden State.

What is a foreclosure?

- Investopedia defines a foreclosure as “the legal process by which a lender seizes and sells a home or property after a borrower is unable to meet their repayment obligation”.
- Essentially, foreclosures are to mortgaged homes as evictions are to rented homes/apartments.
- At the end of 2019, 1 in every 401 housing units were foreclosed in Los Angeles, per a study by Attom.

The Goal of Our Model

- With our model, we'd like to identify macroeconomic trends pertaining to historical unemployment rates, foreclosure statistics, and economic data regarding the general cost of living.
- In identifying these trends via dataset utilization, our model can then predict foreclosure rates in Los Angeles, which can mitigate an issue that has spanned several decades for a major city in California.

FRED®
ECONOMIC DATA | ST. LOUIS FED

COUNTY OF LOS ANGELES
OPEN DATA

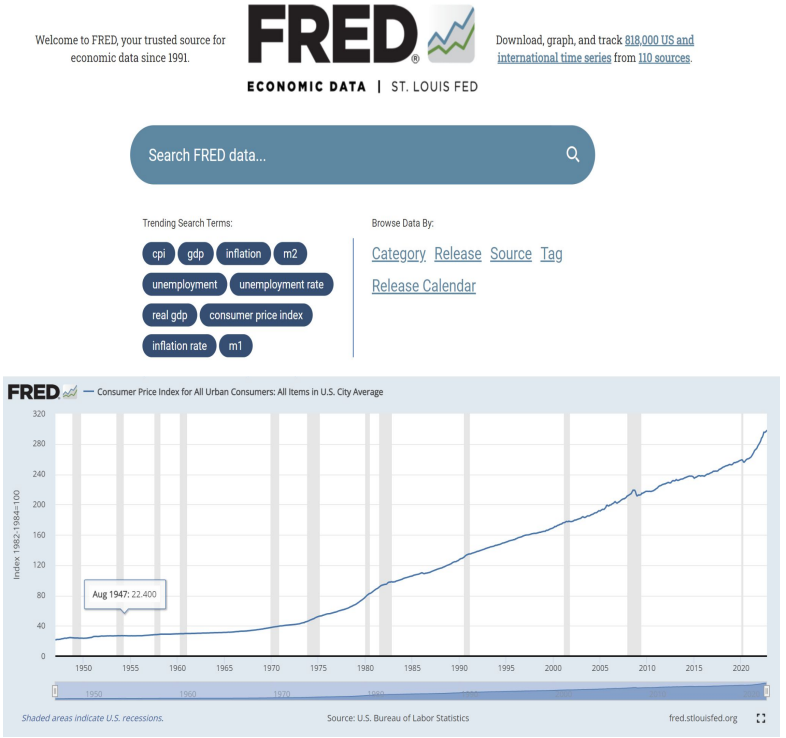
The Tools Used For Our Model

- Institutions such as FRED and LA City's Open Data provided us intuitive ways to search and incorporate economic data.
- Jupyter Notebook allows us to share code and explain our process.
- Git and GitHub helped us maintain version control and distribute changes between team members.
- Python libraries utilized: Pandas, Numpy, SciKit-learn, and Matplotlib.
- Discord was utilized for communication and scheduling amongst team members.

Feature Data

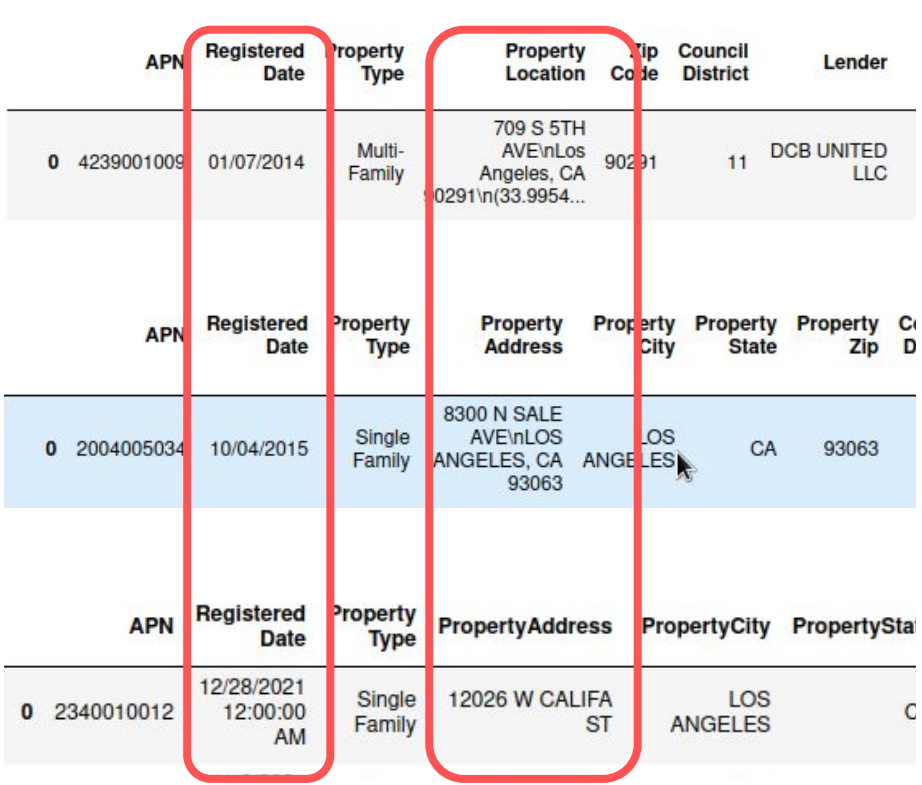
Federal Reserve Economic Data (FRED): An online database containing several types of economic data pertinent to our model. Examples include the Consumer Price Index, Unemployment Rate in California, and Zillow Home Value Index

- Consumer Price Index (CPI): A measure of the average change over time in the prices paid by urban consumers for a market basket of consumer goods and services.
- Unemployment Rate in California: The number of unemployed people in California as a percentage of the labor force.
- The Zillow Home Value Index: A measure of the typical home value and market changes across a given region/housing type (California, in our case).



Data Normalization

Label Data	Feature Data
Date format	Date format
Duplicate Entries	Daily data had to be summed
Non-residential properties	Weekly and irregular data had to be averaged



	APN	Registered Date	Property Type	Property Location	Zip Code	Council District	Lender
0	4239001009	01/07/2014	Multi-Family	709 S 5TH AVE\nLos Angeles, CA 90291\n(33.9954...	90291	11	DCB UNITED LLC

	APN	Registered Date	Property Type	Property Address	Property City	Property State	Property Zip	C D
0	2004005034	10/04/2015	Single Family	8300 N SALE AVE\nLOS ANGELES, CA 93063	LOS ANGELES	CA	93063	

	APN	Registered Date	Property Type	PropertyAddress	PropertyCity	PropertySta
0	2340010012	12/28/2021 12:00:00 AM	Single Family	12026 W CALIFA ST	LOS ANGELES	C

Data Normalization

- Features and Label data were grouped by month
- This was used to join label and feature data
- Ultimately we converted the date to a numerical format (for Matplotlib)

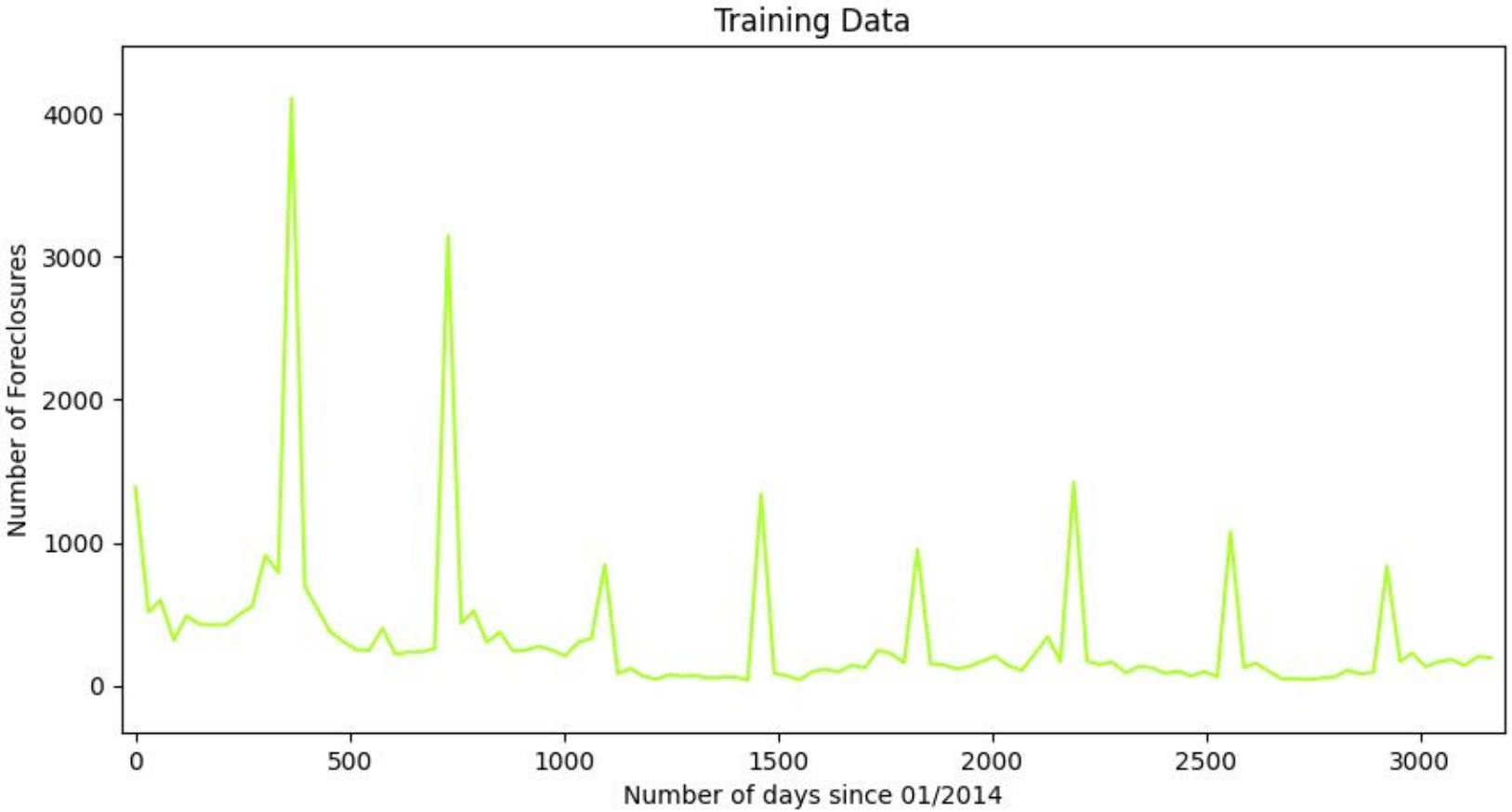
```
def removeTime2021(df):
      X = df
      X['Registered Date'] = df['Registered
Date'].replace(to_replace="\ \d\d\:\d\d\:\d\d\
[A-Z][A-Z]", regex=True, value="")
      return X

def innerJoinOnDATE(*argv):
    if len(argv) < 2:
        raise Exception("You will need at least two
dataframe to merge.")
    df = pd.merge(argv[0], argv[1], on='DATE')
    for i in range(2, len(argv)):
        df = pd.merge(df, argv[i], on='DATE')
    DATE = df.pop('DATE')
    df.insert(0, 'DATE', DATE)
    return df
```

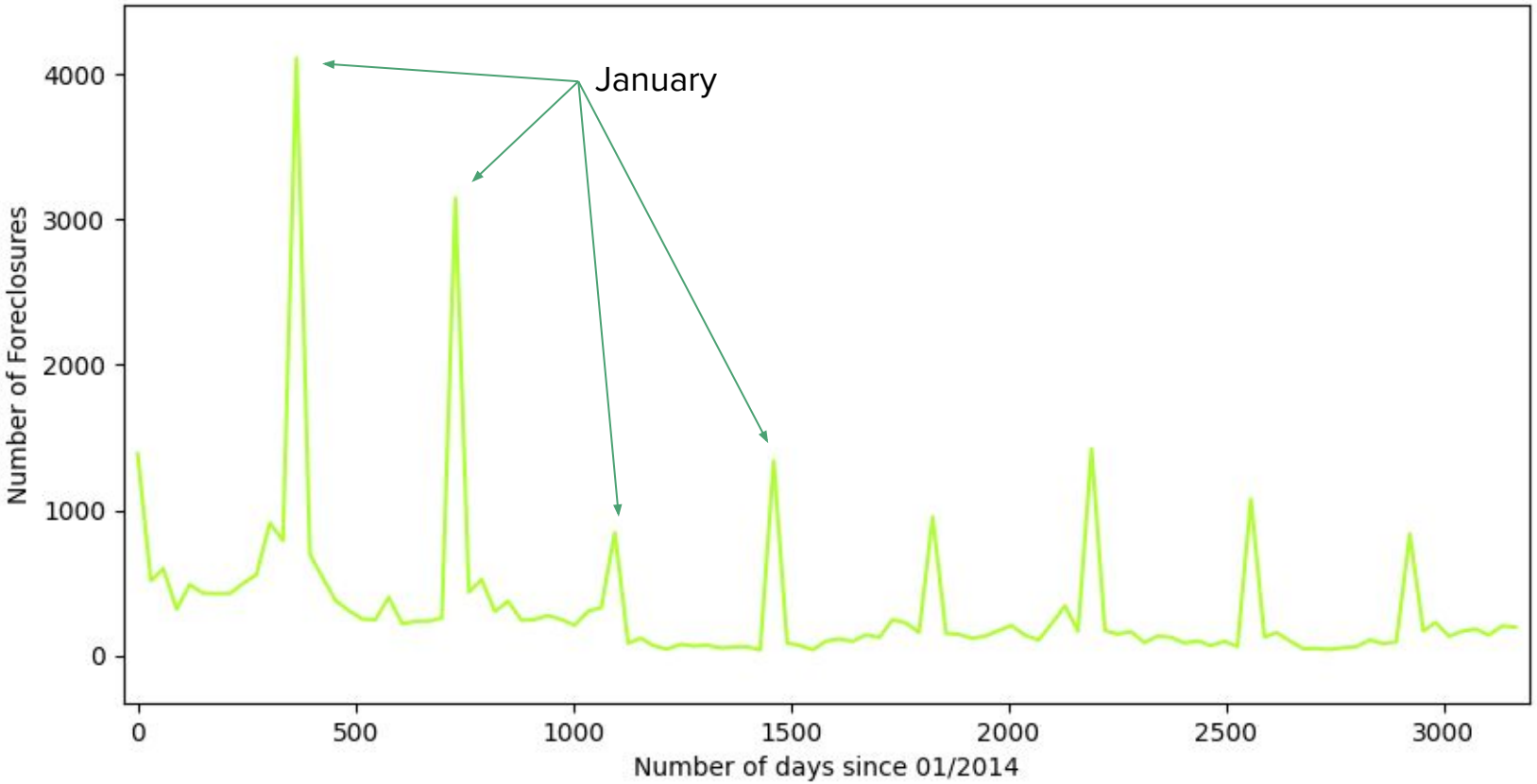
Combined Feature and Label Data

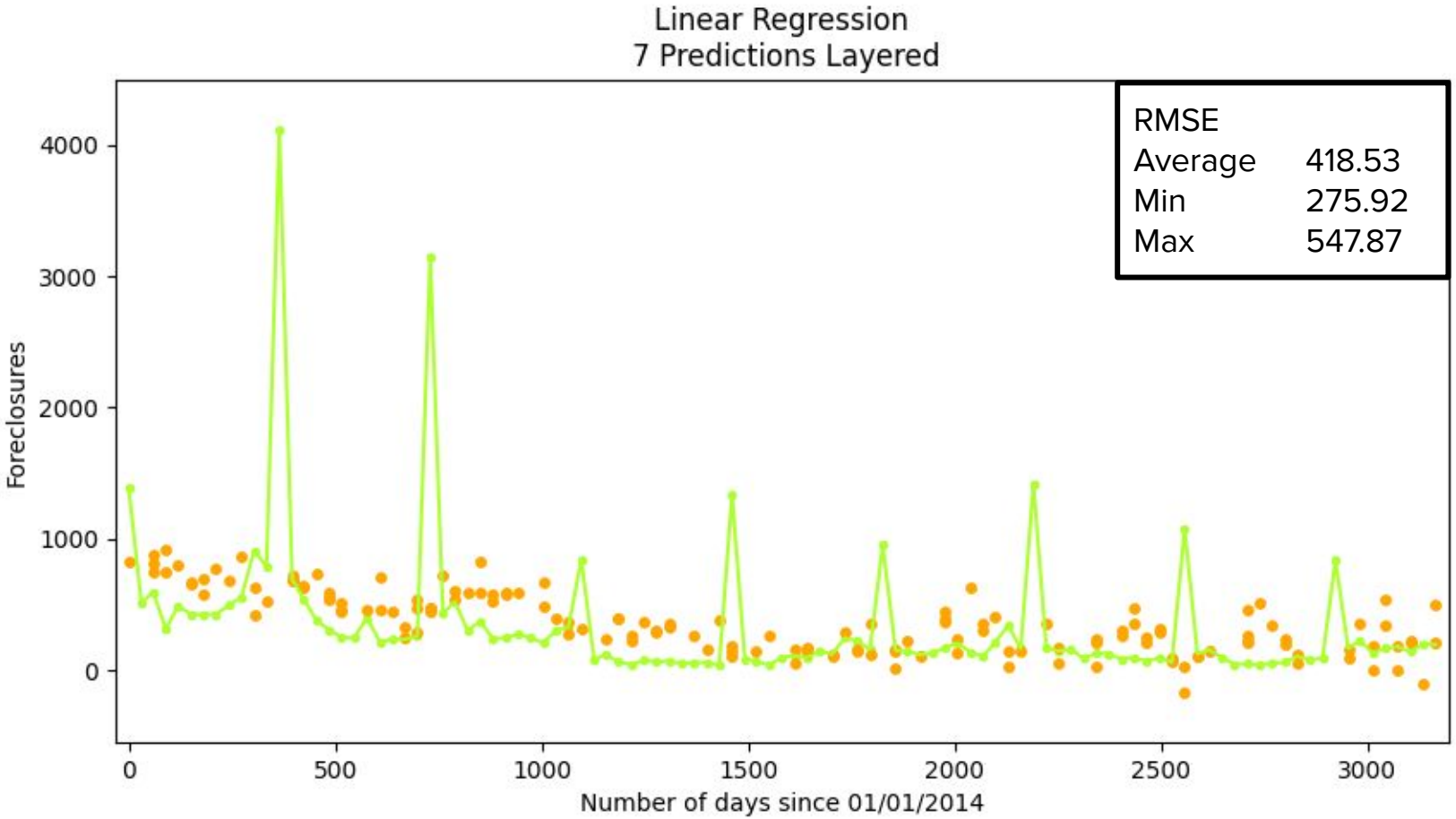
Total Foreclosures: 35874

Total Entries in DataFrame: 106 (Months from 01/2014 to 09/22)

	DATE	FORECLOSURE	CAUR	Average DGS	DFF	NASDAQCOM	ZHVI	MORT30US	BofA_Yield_Index_PCH	CA_Consumer_Price_Index	CPI STICKY
0	2014-01-01	1388	8.2	2.858095	0.071613	1.92538	1.34214	4.4320	1.07487	235.288	1.821814
1	2014-02-01	513	8.1	2.709474	0.066429	1.08537	1.13758	4.3025	0.87364	235.547	1.789186
2	2014-03-01	597	8.0	2.723333	0.078065	1.83138	0.60021	4.3425	1.04621	236.028	1.820857
3	2014-04-01	317	7.8	2.705238	0.090333	-3.67258	0.22288	4.3375	0.66026	236.468	2.025161
4	2014-05-01	488	7.7	2.559048	0.087097	0.39007	0.08087	4.1920	0.88867	236.918	2.128235
5	2014-06-01	427	7.6	2.598571	0.095667	4.77259	0.13118	4.1625	0.99437	237.231	2.143341
6	2014-07-01	425	7.5	2.542273	0.090968	2.34003	0.11132	4.1300	-0.00757	237.498	2.100809
7	2014-08-01	426	7.4	2.420000	0.088065	0.69248	-0.04856	4.1150	-0.27603	237.460	1.962504
8	2014-09-01	496	7.3	2.534286	0.088333	1.94307	0.16591	4.1625	-0.25902	237.477	1.954047
9	2014-10-01	556	7.2	2.304091	0.089032	-3.25943	0.38947	4.0360	-0.67733	237.430	2.006019
10	2014-11-01	911	7.1	2.325556	0.092000	6.46052	0.65561	3.9975	0.25513	236.983	2.004366
11	2014-12-01	790	6.9	2.207273	0.122581	0.95994	0.58631	3.8640	-2.15708	236.252	1.971126
12	2015-01-01	4109	6.8	1.881500	0.114839	-1.24664	0.61772	3.6700	0.61587	234.747	1.985653
13	2015-02-01	692	6.7	1.975263	0.110714	3.86320	0.65590	3.7100	1.83100	235.342	1.981997
14	2015-03-01	543	6.6	2.042727	0.112903	1.72540	0.70982	3.7700	0.33258	235.976	1.999137


Training Data
Number of Foreclosures
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
Number of days since 01/2014

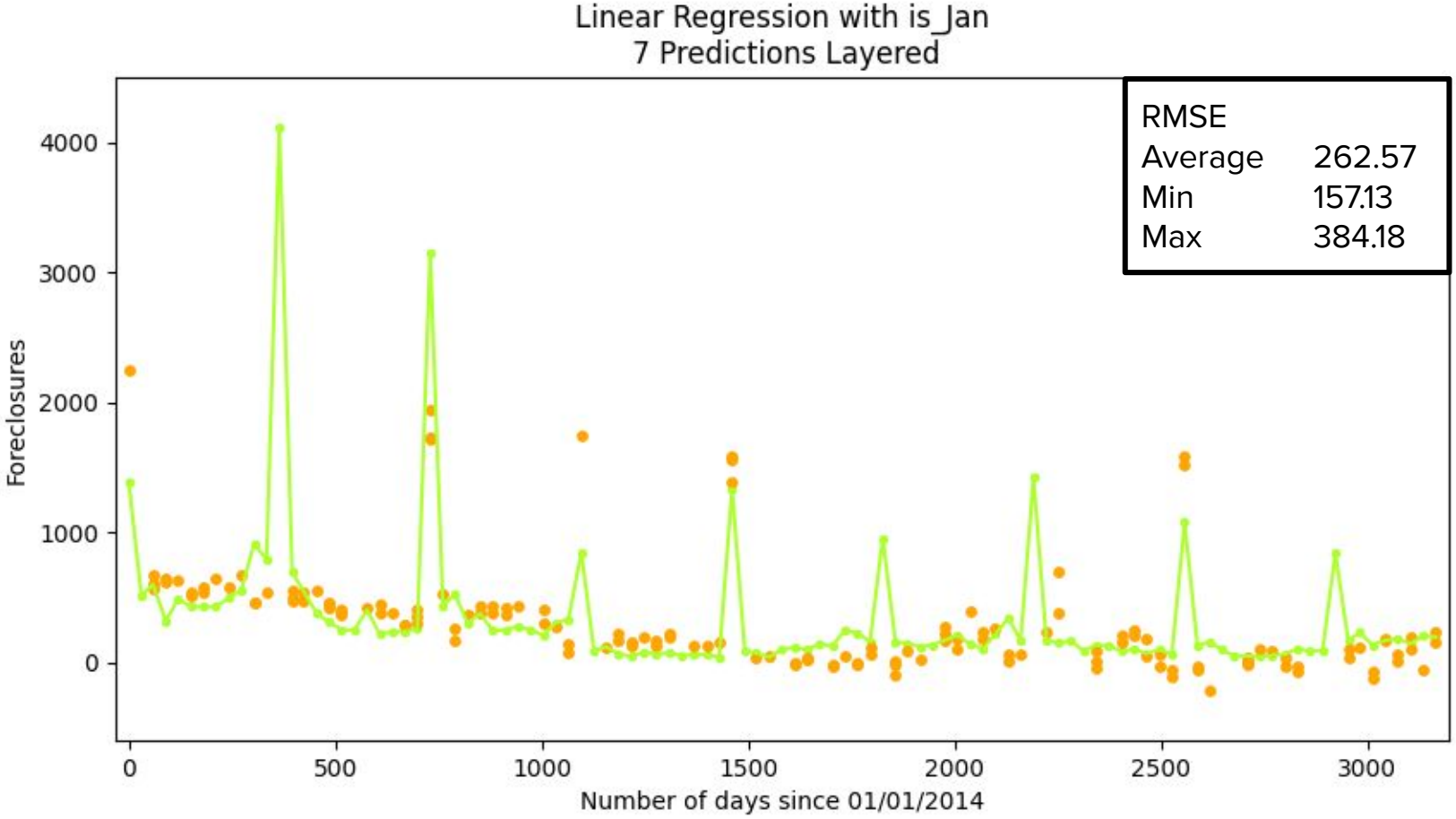

January
Number of Foreclosures
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
Number of days since 01/2014


Linear Regression
7 Predictions Layered
RMSE
Average 418.53
Min 275.92
Max 547.87
Foreclosures
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
Number of days since 01/01/2014

Improving Prediction: Classification Feature

- Wang came up with classifying each month of January
- Date feature was converted to datetime type, values are: 0 to 3165
- Data held in Months/Years lost

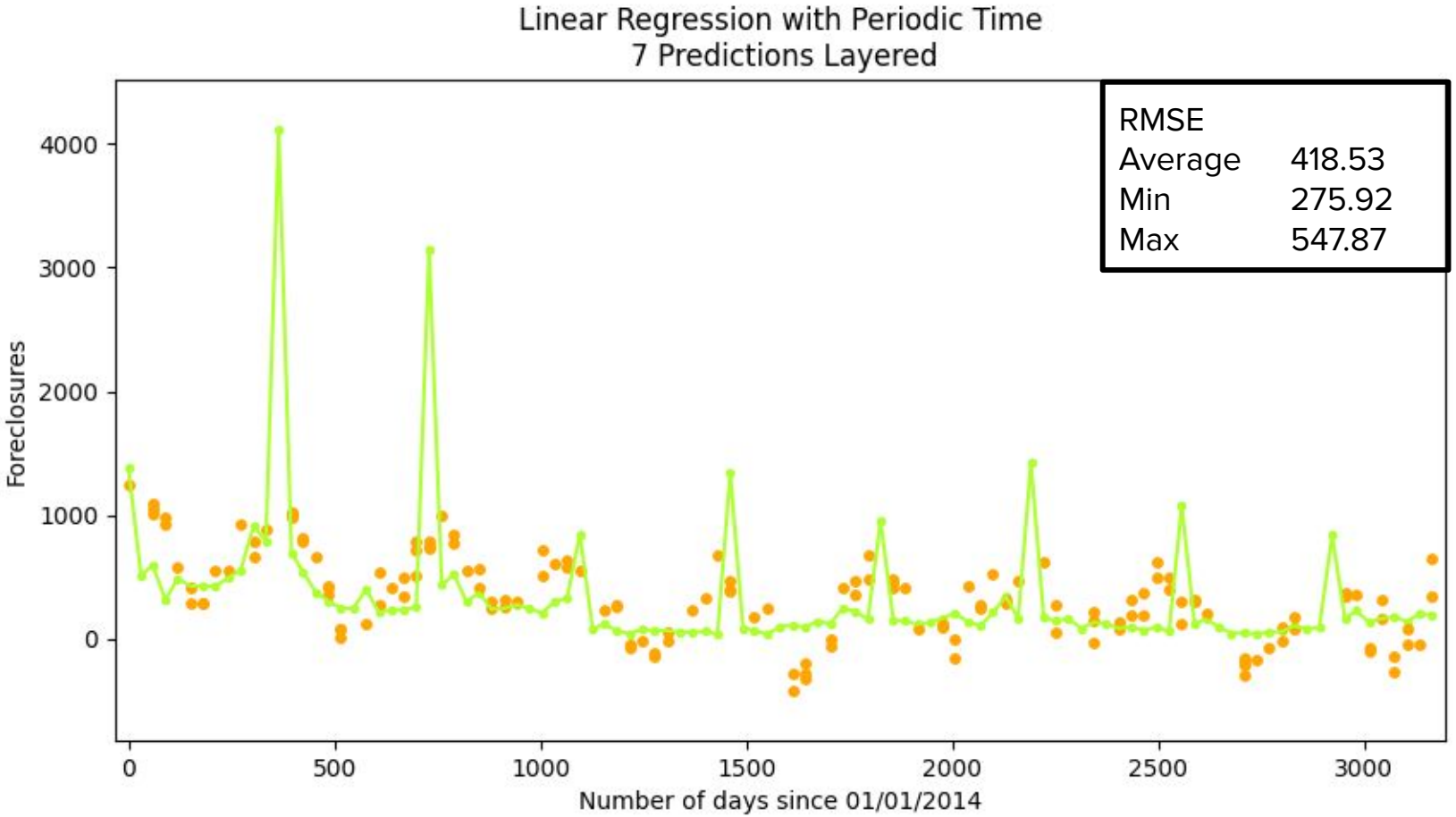
```
def isJan(date):
     if date[5:7] == '01':
          return 1
     else:
          return 0

df['isJan'] = df['DATE'].apply(lambda x:
isJan(x))
```

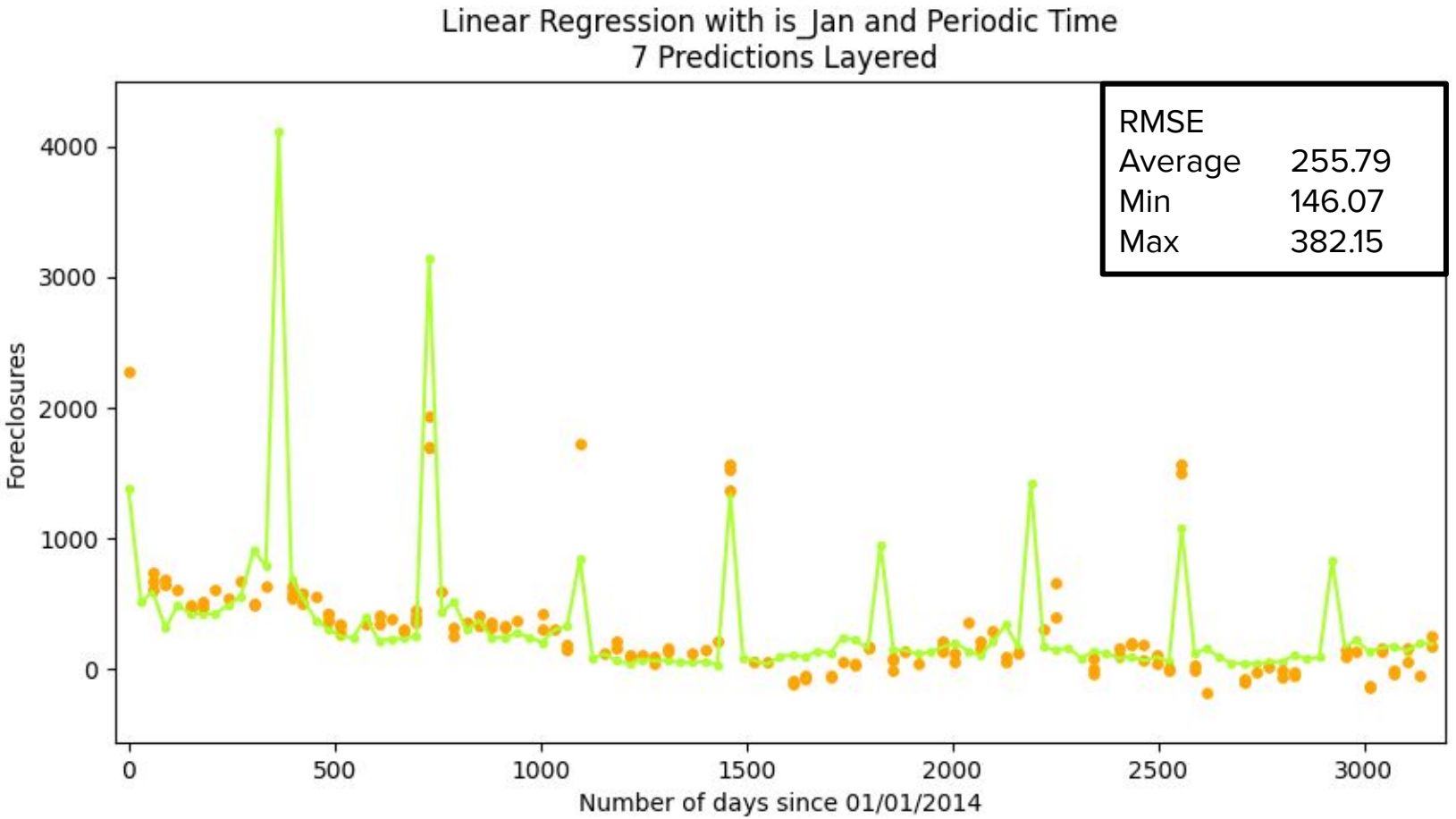

Linear Regression with is_Jan
7 Predictions Layered
RMSE
Average 262.57
Min 157.13
Max 384.18
Foreclosures
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
Number of days since 01/01/2014

Improving Model: Cyclical Features

- Cyclical Features like time: Months, Years, Hours benefit by being mapped to a periodic function.
- This is more digestible for the regression algorithm

```
df['mnth_sin'] = df['DATE'].apply(lambda x:
np.sin((int(x[5:7])-1)*(2.*np.pi/12)))

df['mnth_cos'] = df['DATE'].apply(lambda x:
np.cos((int(x[5:7])-1)*(2.*np.pi/12)))
```

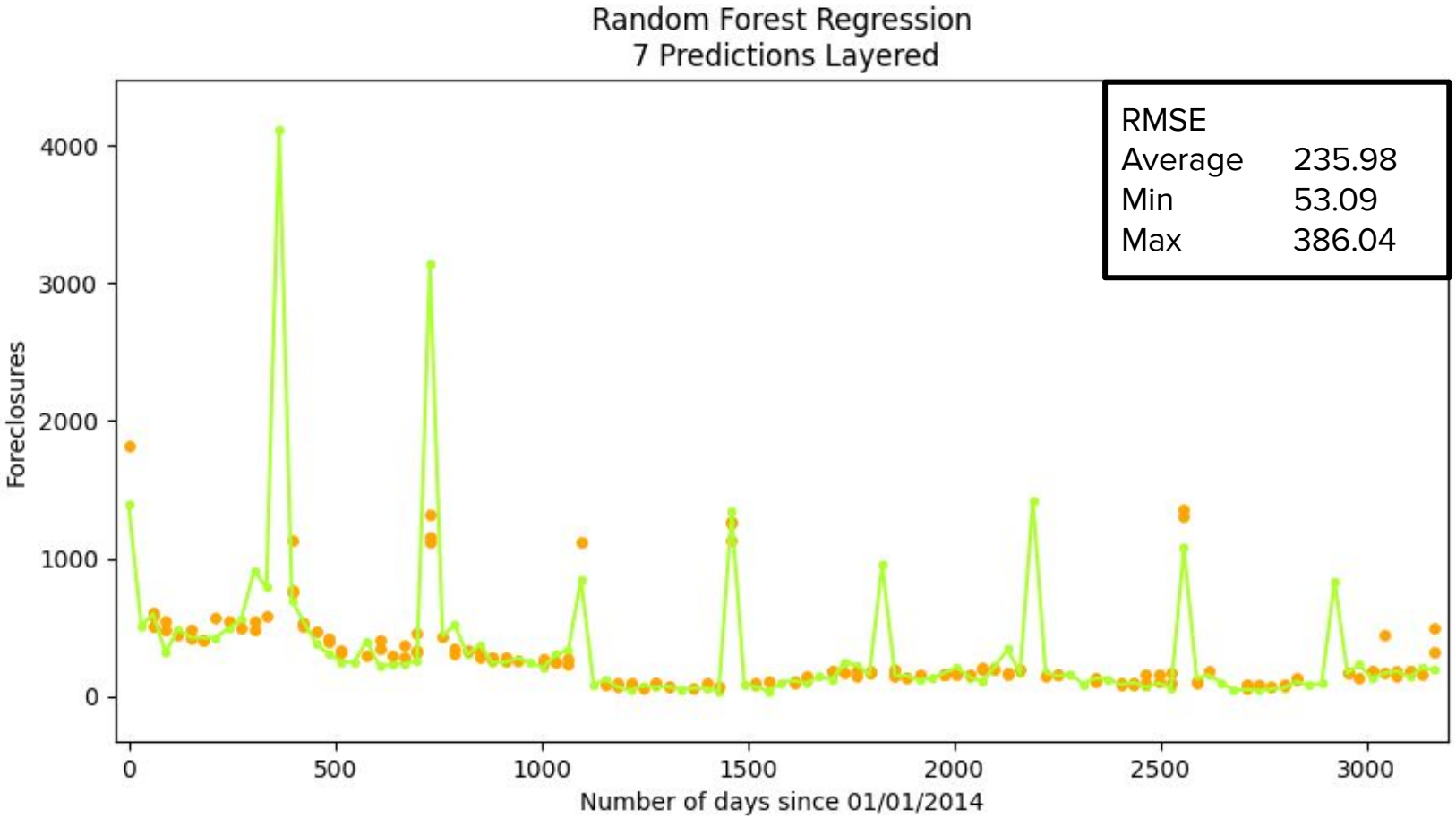

Linear Regression with Periodic Time
7 Predictions Layered
RMSE
Average 418.53
Min 275.92
Max 547.87
Foreclosures
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
Number of days since 01/01/2014


Linear Regression with is_Jan and Periodic Time
7 Predictions Layered
RMSE
Average 255.79
Min 146.07
Max 382.15
Foreclosures
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
Number of days since 01/01/2014

Training and Testing Different Splits

- Train Test Splits effect prediction
- January foreclosures jump by an order of magnitude

```
### CODE TO TEST OUT DIFFERENT SPLITS
for i in range(0,20):
      X_train, X_test_3, y_train, y_test = train_test_split(X, y, test_size=0.23, random_state=i)
      regressor = LinearRegression()
      regressor.fit(X_train,y_train)
      y_pred_3 = regressor.predict(X_test_3)
      mse = metrics.mean_squared_error(y_test, y_pred_3)
      rmse = np.sqrt(mse)
      print(f"R_2: {r2_score(y_test,y_pred_3)}, RMSE {rmse} i {i}")
      avg_rmse += rmse
      avg_r2 += abs(r2_score(y_test,y_pred_3))
      plt.scatter(X_test_3['DATE_DELTA'], y_pred_3,s=15, c="orange")
```


Random Forest Regression
7 Predictions Layered
RMSE
Average 235.98
Min 53.09
Max 386.04
Foreclosures
0
1000
2000
3000
4000
0
500
1000
1500
2000
2500
3000
Number of days since 01/01/2014

Conclusion

What we can improve:

- More training / label data
- New Algorithms
- Data local to Los Angeles
- Model may be over fitted for data

What can we do with what we have:

- This model can be used with different economic data
- Could be used as boiler plate code for another project

Q & A