San Diego House Value Analysis

Advisor: Prof Ilkay Altintas, Prof Volkan Vural
Team Members: Shammi Khattar, Serdar Begnazarov, Amer Catovic, Hessam Jalali Ghajar
MAS DSE C4, 2019 - Group 2

Abstract

The general business challenge that this capstone project aims to address is to evaluate and predict housing values in San Diego County. The intended public benefit from the project is to provide further insights and guidance to homeowners, investors and government sectors. The last year's cohort established a baseline for our work. Our task was to provide additional and enhanced results and insights using the following three avenues: new features, enhanced model and scalable architecture. The specific objective we set at the beginning was to improve the RMSE by at least 10% with respect to the baseline. Our results indicate improvement of 34%.

1. Introduction and Question Formulation

Our capstone project aims to evaluate and predict housing values in San Diego County more accurately than the baseline. Our task is to provide additional and enhanced insights. Several important factors that determine the property values, such as regionality (coastal, inland.), the proximity of schools, seasonality , house size and stability in the San Diego County are built in the baseline model.

We intend to expand this to include:

- Macroeconomic factors: mortgage rates, GDP
- Microeconomic factors, such as employment and distance to Coast
- (Hybrid) Regional factors like Crime rates and statistics, School districts and their Ratings

Our primary **goal** is to improve the housing price forecast model and hence the accuracy of price prediction by **10%** over baseline for the San Diego County.

The main challenge is to get public data for the features that contribute towards housing prices. For example, one would imagine remodeling data would be useful in predicting a property price. However that data is not available. Our project, however used data from the US Census, San Diego County, SANDAG and SchoolDigger. The process of cleansing and loading even though tedious is the first step towards EDA and extract good features to be fed to any ML algorithms.

Our intention is to answer the following questions:

a) How important is proximity to the local employment hubs?
b) Do prices vary more or less for various property segment types (SFR, MFR, Condos)?
c) Does local crime in zip code have any effect on pricing?
d) What features are most important outside of traditional characteristics when someone buys a property?
e) For places like San Diego, Do people give much importance to buying a property closer to the coast or be in a better school district?

After data was preprocessed, cleaned and EDA was performed, our models found that when crime data was available people did give Crime more importance. People like to buy closer to the coast, however this was more apparent for Condos than other property segment types. The next few sections of this report detail more analysis and insights into key findings.

2. Team Roles and Responsibilities

Project Management: Shammi is responsible for project coordination and management. He is also responsible for database management and backups, database deployment, integration and querying.
Shammi organizes meetings within the team and with advisors, creates plans with team members for each step and to ensure goals are accomplished on time. Shammi worked on mortgage and migration features and contributed to all deliverables during the project lifecycle.

Feature engineering and general data engineering function: Amer took the responsibility for feature engineering role as well as the majority of all other data engineering related tasks.

Modeling / Features / Kubernetes / Git:
Serdar is the lead engineer on the project. He focused on modeling and machine learning tasks. He performed model selections and different ML techniques such as segmentation, clustering, anomaly removal, etc. He also developed massively parallel hyperparameter tuning pipeline on Kubernetes cluster.
Serdar also contributed on Feature Engineering/Augmentation. He worked on adjusting price related feature to inflation. He also acquired Economic Conditions Index. He augmented some features from existing ones.
Serdar benchmarked the resources allocated for `words-housing` namespace on Kubernetes cluster and communicated with the cluster admins for additional resources and modules to complete the project on timely manner.

Serdar managed the mirroring of source code from GitHub to GitLab and vice versa for deployment of parallel jobs.

Visualization: Hessam is leading this role throughout the project. He developed visualization tools and dashboards. He developed tools for final presentation demo as well. He also acquired crime data, preprocessed, and introduced new engineered features for the model.

3. Data Acquisition

Data Sources, Data Sets and Size

Dataset Sources	Data Linkage	Destination Pipeline	Scripts	Data Size	Records
County - Property Characteristics (PAR)	https://drive.google.com/drive/u/0/folders/1kf9h_wCDUWtVUFd2VwWGhYkE8VocDkvk	Characteristics (PAR)	SQL Server ETL packages	1 GB/month We will probably have 15 years worth of data load	1.3 million/file
County - Sales	https://drive.google.com/drive/u/0/folders/1kf9h_wCDUWtVUFd2VwWGhYkE8VocDkvk	Sales	SQL Server ETL packages	30 MB/month. Data available since 1983	2.5 millions records total
County - MPR	https://drive.google.com/drive/u/0/folders/1kf9h_wCDUWtVUFd2VwWGhYkE8VocDkvk	(Property Value) MPR	SQL Server ETL packages	38 MB/month We will probably have 15 years worth of data load	1.3 millions
County - Full and Detailed Address	https://drive.google.com/drive/u/0/folders/1kf9h_wCDUWtVUFd2VwWGhYkE8VocDkvk	Clean data set for property addresses	SQL Server ETL packages	186 MB	1.4 millions
Crime	https://www.sandag.org/index.asp?classid=14&subclassid=21&projectid=446&fuseaction=projects.detail	Crime	SQL Server ETL packages	132 MB Jan 1st 2012 to Dec 31st 2017	160,000 per year
Mortgage Rates*	https://fred.stlouisfed.org/series/MORTGAGE30US	Mortgage rates (30FRM, 15FRM, 5/1 ARMs after 2005)	SQL Server ETL packages Python notebooks	135 KB (1971-2018)	2500

San Diego Housing Price Index	https://fred.stlouisfed.org/series/SDXRSA	HPI (Monthly)	Python Notebooks	12 KB (1987-2019)	387
San Diego Consumer Price Index	https://data.bls.gov/timeseries/CUURS49ESA0	CPI (Annual)	Python Notebooks	8 KB (1965-2019)	55
San Diego Economic Conditions Index	https://fred.stlouisfed.org/series/SDIAGRIDX	ECI (Monthly)	Python Notebooks	8KB (1990-2019)	350
School Ratings	https://drive.google.com/drive/folders/1xeQRGLobT6M8kZxjZPAS4SB0HwI8b6ku	School ratings	Python Notebooks	60MB (2018)	700,000
US GDP	https://www.multpl.com/us-real-gdp-growth-rate/table/by-quarter	GDP change (quarterly)	Python notebooks	2kB (1988-2018)	130
UC Census Business Patterns	https://census.gov/programs-surveys/cbp/data/datasets.html	Employment hubs (annual, per zipcode)	Python Notebooks, SQL scripts	64kB(2004-2016)	1600
Coastline points	Manual	Coast distance	SQL scripts	2kB	40

Table 1: Data source description

Data Ingestion

Based on the schema of incoming data, various tables within the postgres DB were created. The data was primarily loaded using Python and DB libraries like sqlalchemy.

The incoming data was in CSV and JSON formats, both were loaded in the DB easily. Histograms and scatter plots were used to find any obvious anomalies. If there were any NaNs found in the data - those records were dropped.

Data Exploration Summary for Key Features

Employment (Microeconomic):

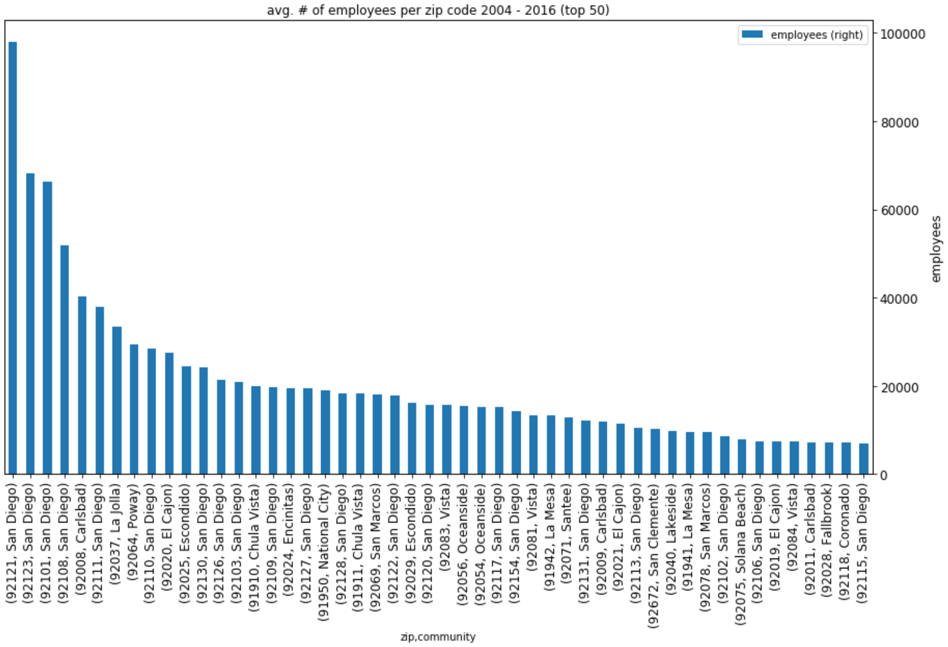
Employment data is formatted in yearly files in the csv format, with one row for each zip code and 10-12 columns for the features. The following steps were performed with this data:

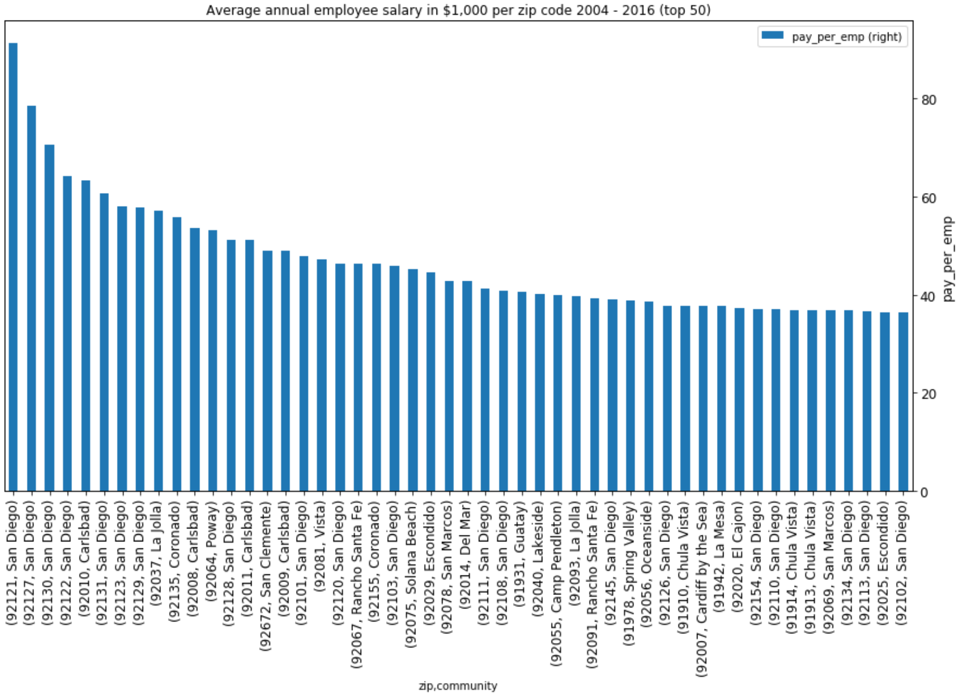
1. Staging (pre-processing)

   Python NB was created for merging the yearly csv data files into a single csv file. The NB includes the following steps:
   - Remove inconsistency in the number of features and their naming over the years. This was fixed and a set of features was selected for the merged file.
   - Filter the San Diego County zip codes based on the zip_codes table in the database
   - Save the merged and staged data as a new csv file, to be imported into the database
2. Database import
   - The employment data in the csv format from step 1 was imported into the database as a new table zipcode_employment
3. EDA

   The following EDA steps were performed using python notebook:
   a. Load employment data from the database into a pandas dataframe
   b. Clean up the data by interpolating the null values. These values occur when the number of companies is very small, so the aggregate salary data may reveal confidential information, so such data was removed by the US Census.
   c. Compute the payroll per establishment (pay_per_est) and salary per employee (pay_per_emp) and add them as new columns in the database. The salaries are expected to be among the most relevant features related to employment.
   d. Generate the histograms of employment data to understand the ranges and the zip codes that are at each end of the ranges.

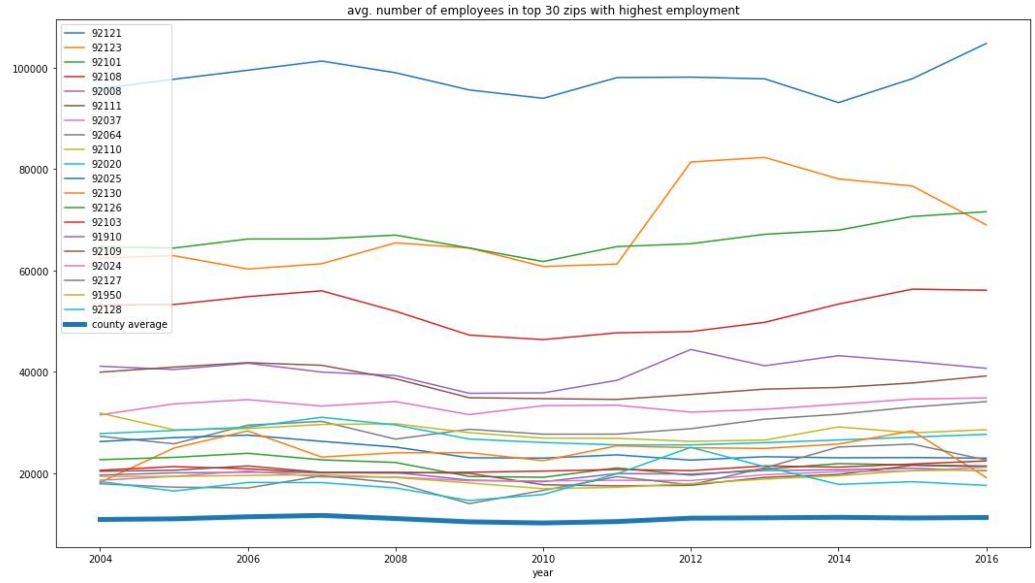
      First, we look at the histogram of the total number of employed people per zip code, averaged over the 13 years. This should have some correlation to the overall purchasing power and population density. We see that there is a wide range of values, which indicates that there is a variety of business and population densities in the county.

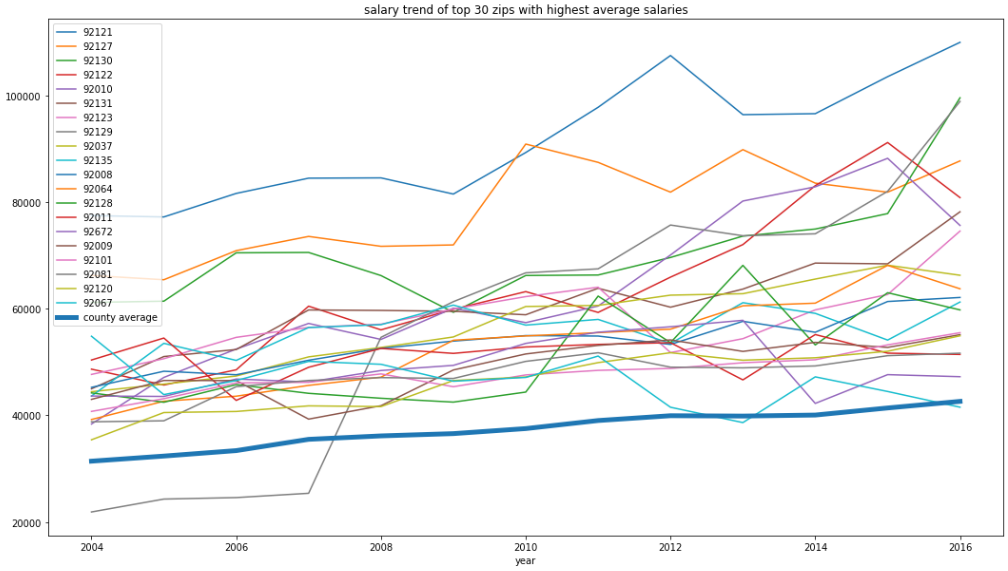

avg. # of employees per zip code 2004 - 2016 (top 50)


Average annual employee salary in $1,000 per zip code 2004 - 2016 (top 50)

e. Next, we can see the histograms of annual salaries, over the 2004-2016 period. Even a superficial examination reveals that some of the zip codes (e.g. 92121 and 92037) have both among the highest number of employees and the highest salaries. We would expect this to correlate strongly with the house prices in some way.

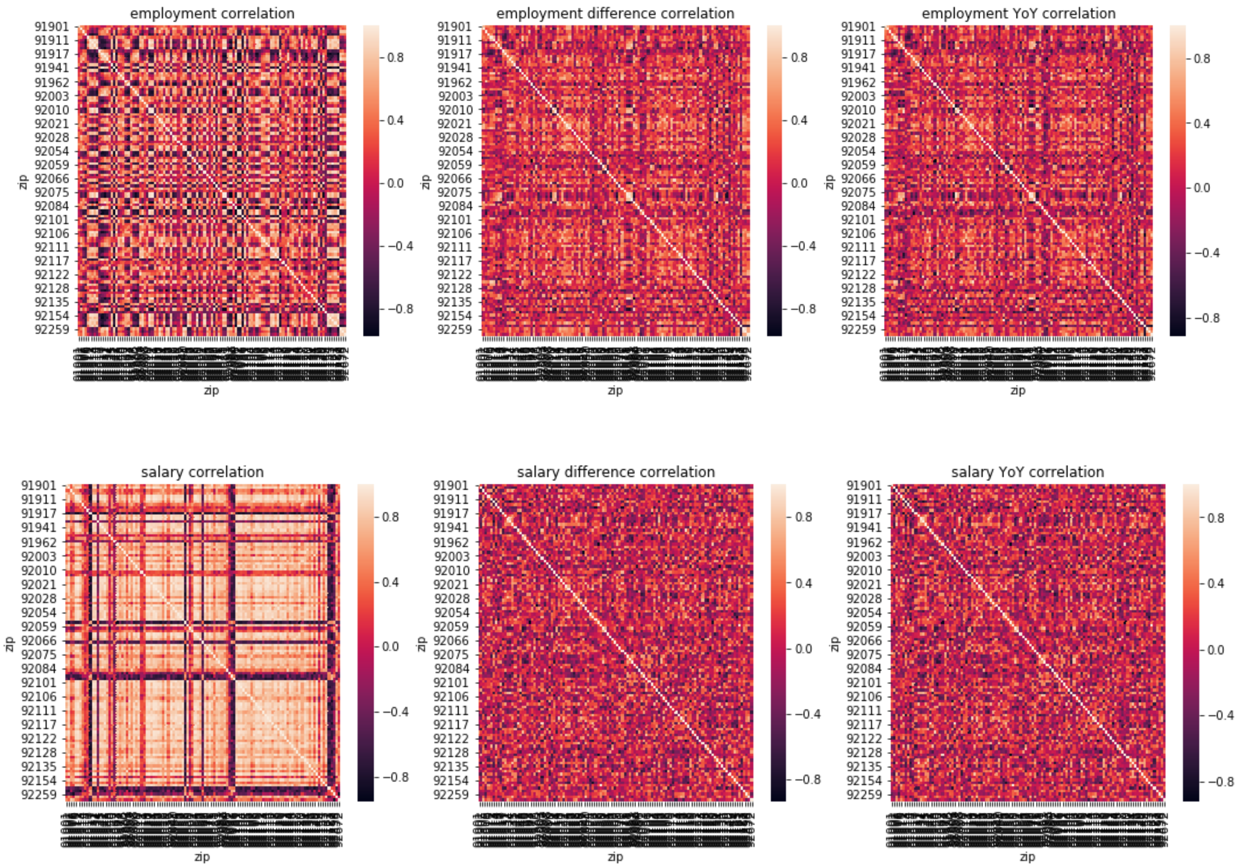
f. Next, we looked at the trends over the 2004-2016 period. We saw a somewhat surprisingly even employment over the years, with a lull during the 2009 recession. It appears though that the zip codes with high employment have a more upward trend in recent years compared to other zip codes.





The salary trend is shown next. Here again, it seems that high income area have stronger upward trend in recent years than the areas with lower incomes.

g. In the final step, we looked at the correlation between the zip codes in terms of the employment and salaries. This may be the most interesting plot of all: it shows that many zip codes have negative correlations in terms of employment and salaries, in some cases quite strong. While this may be an interesting observation, it remains to be seen if this has any implications on the housing prices per se, as opposed to, for example pure economic impacts (e.g. competing companies/industries etc.).



Mortgage (Macroeconomic)

1. Data from St Louis Fed was provided in the form of raw CSV flat files with 9 fields.
2. For EDA, we focussed on 30 YR FRM and 15 YR FRM. The rates are not for San Diego country but average rates that Fed is tracking for the entire nation along with the points charged for taking a loan. The data does not assume any specific down payment percentage.

The following table provides the set of attributes analyzed for exploration:

30 YR Fixed Rate Mortgage APR
30 YR Points
15 YR Fixed Rate Mortgage APR
15 YR Points
5/1 ARM APR
5/1 ARM Points

Key Findings of EDA

1. **Employment:** There is a significant degree of variation of the employment and salaries both geographically (from one zip code to another) and over time. In terms of geographical variation, some zip codes are strongly ahead of the pack in terms of the number of employees and salaries. It also appears that the high performing zip codes have registered steeper growth over time. An interesting initial observation is that there is a strong negative correlation between some zip codes in terms of employment and salaries. It remains to be seen whether there is any causality here. Another key challenge will be to establish relationship between the geographical aspect of employment and salaries, and housing prices. On the other hand, the hope is that the temporal relationship between housing prices and the aggregate county-level employment would be easier to establish.

2. **Mortgage Rates**: Historically there has been a strong correlation between mortgage rates and the growth observed in the year to year property prices. The following plots illustrate the sold_price vs sold year. If we were to correlate the price upward or downward trend, its strongly observed when the rates are lower. The 30 YR FRM APR has been on the downward trend since 1981. Looking at the average property prices in San Diego county during the same temporal range overall the prices have gone up, with the exception of 2008-2010 recession. That seems to be attributed more towards the onset of 3/1 && 5/1 ARMs.

3. **Distance to Coast:** We selected 40 points on the SD county coastline. For each property, we computed the minimum distance to the 40 coastline points and used as the distance to coast for that property.

4. **School Ratings:** We used the www.schooldigger.com API to obtain the school districts, elementary and secondary, serving each property and their ratings. If a property was part

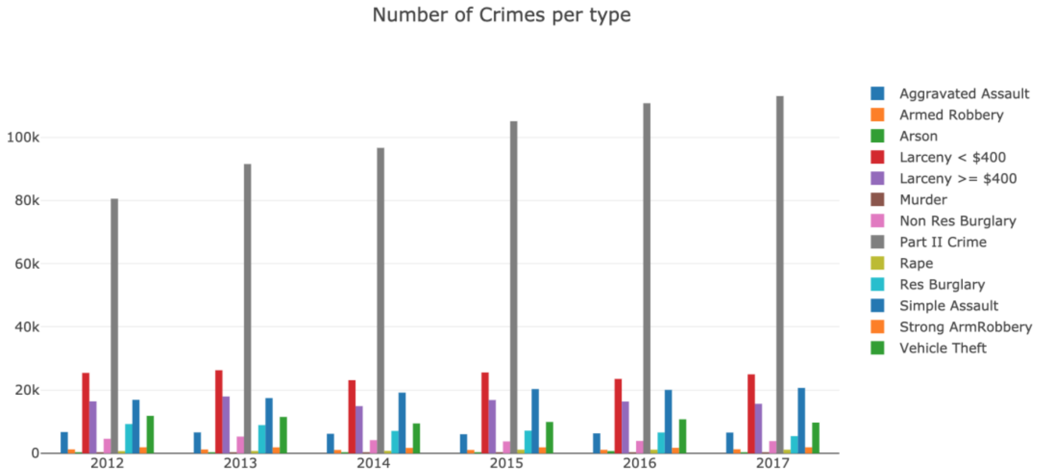
of a unified school district, we would then consider the district as both the elementary and secondary school district. This data was available for the year 2018 only. This feature was considered as linked to the property.

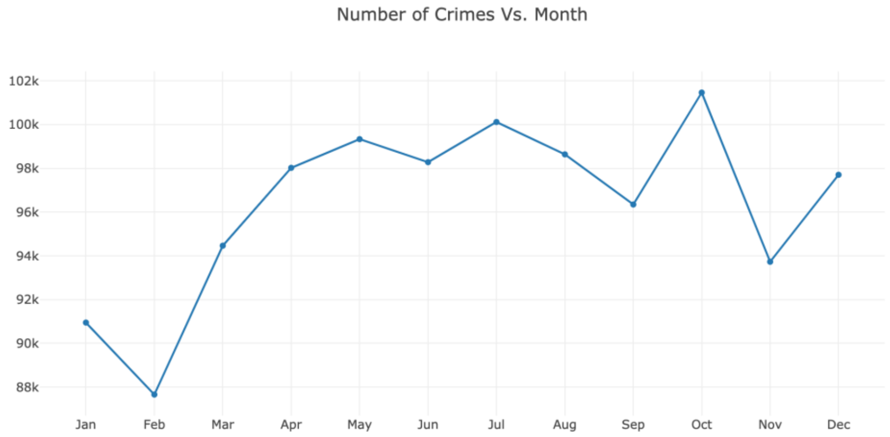
5. **GDP growth:** We used the data for the quarterly GDP growth. This feature was considered as linked to the property transaction.

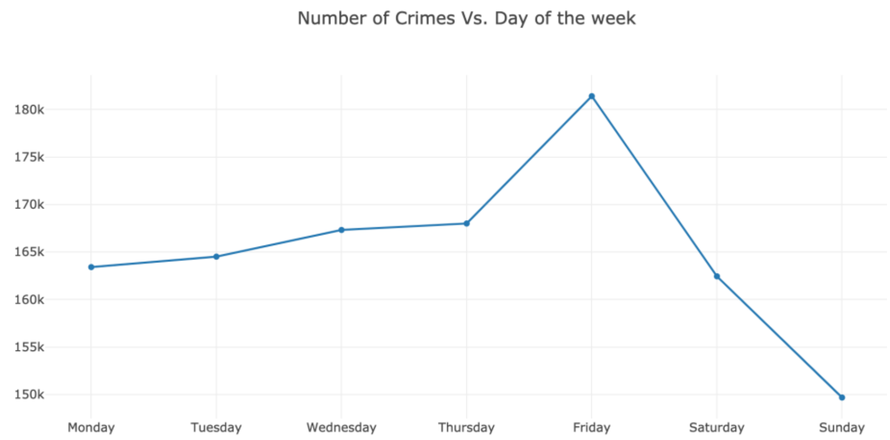
6. **Crime Data Exploration:**
   Raw data description:
   - Dataset includes geocoded crime incidents from 2012 to December 31st of 2017 that were provided by the San Diego Association of Governments (SANDAG). The crime data is updated daily by different agencies throughout San Diego county. In the crime data, there are 795k total data incidents starting from 183k in 2012 to 210k in 2017 per year. Address block field is only provided to the nearest hundred block in order to maintain privacy, so we decided to use zip code column as a primary feature for the choropleth map.

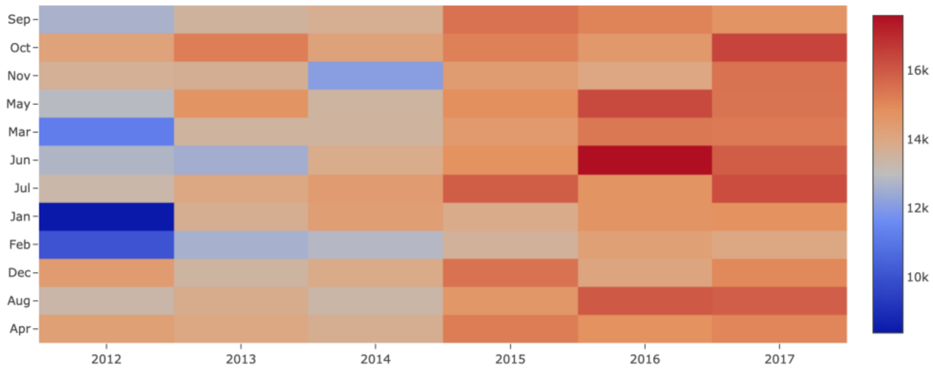
   Data processing:
   - The schema from 2012-2015 was different from 2016-2017, so some matching had to be done before merging all the data together. Zip code, and the city name have some missing values as well as some mixed values. For example, there are some zip code value in a city name and vice versa that made the preprocessing complicated.

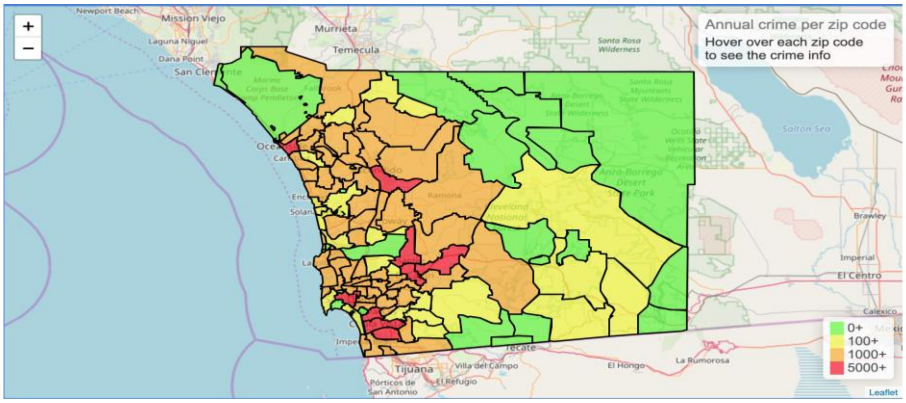
   Some EDA graphs to explore the dataset:


Number of Crimes per type
100k
80k
60k
40k
20k
0
2012
2013
2014
2015
2016
2017
Aggravated Assault
Armed Robbery
Arson
Larceny < $400
Larceny >= $400
Murder
Non Res Burglary
Part II Crime
Rape
Res Burglary
Simple Assault
Strong ArmRobbery
Vehicle Theft


Number of Crimes Vs. Month
102k
100k
98k
96k
94k
92k
90k
88k
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Number of Crimes Vs. Day of the week




Annual crime per zip code
Hover over each zip code to see the crime info

1. San Diego (92101):

Total:9504

Top 5:

Category	Count
Part II Crime	4504
Simple Assault	1527
Larceny < $400	1101
Larceny >= $400	903
Aggravated Assault	537

2. Oceanside (92054):

Total:7612

Top 5:

Category	Count
Part II Crime	4826
Larceny < $400	948
Simple Assault	571
Larceny >= $400	528
Vehicle Theft	190

3. El Cajon (92020):

Total:6867

Top 5:

Category	Count
Part II Crime	4609
Larceny < $400	630
Simple Assault	479
Larceny >= $400	325
Vehicle Theft	294

Data pipeline

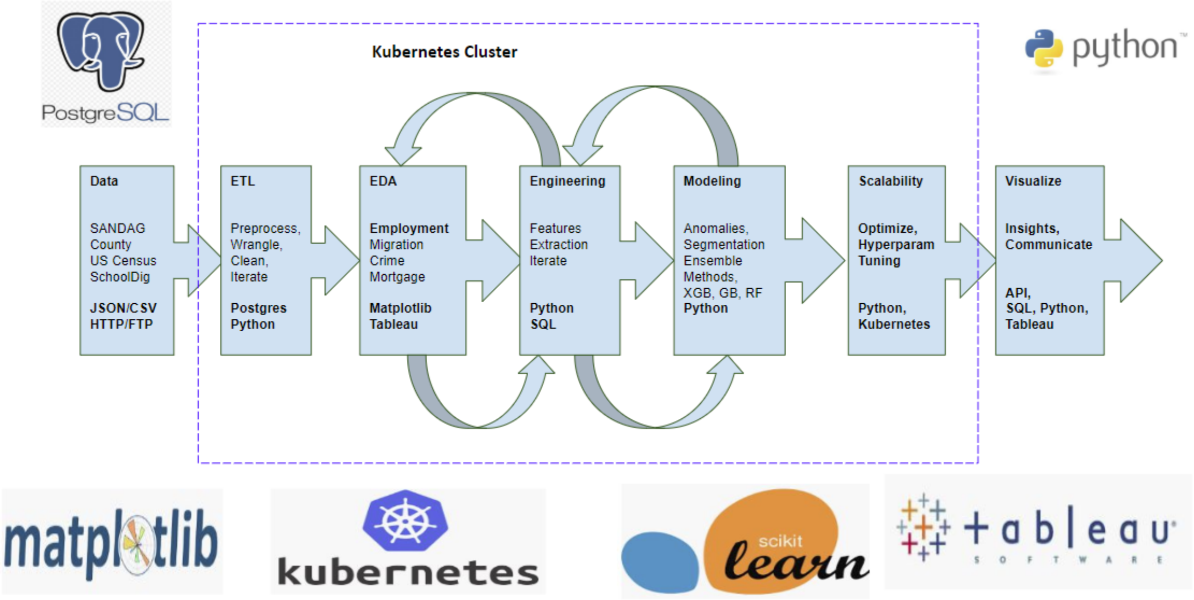


Table: Pipeline

The data pipelines above shows at a high level the data flowed from the source into our database which was hosted as part of the Kubernetes cluster. Once the data was cleaned and preprocessed, EDA was performed. Once some insights were projected using various plotting libraries and tools like Tableau - features were extracted, engineered and further cleaned before being fed to ML models.

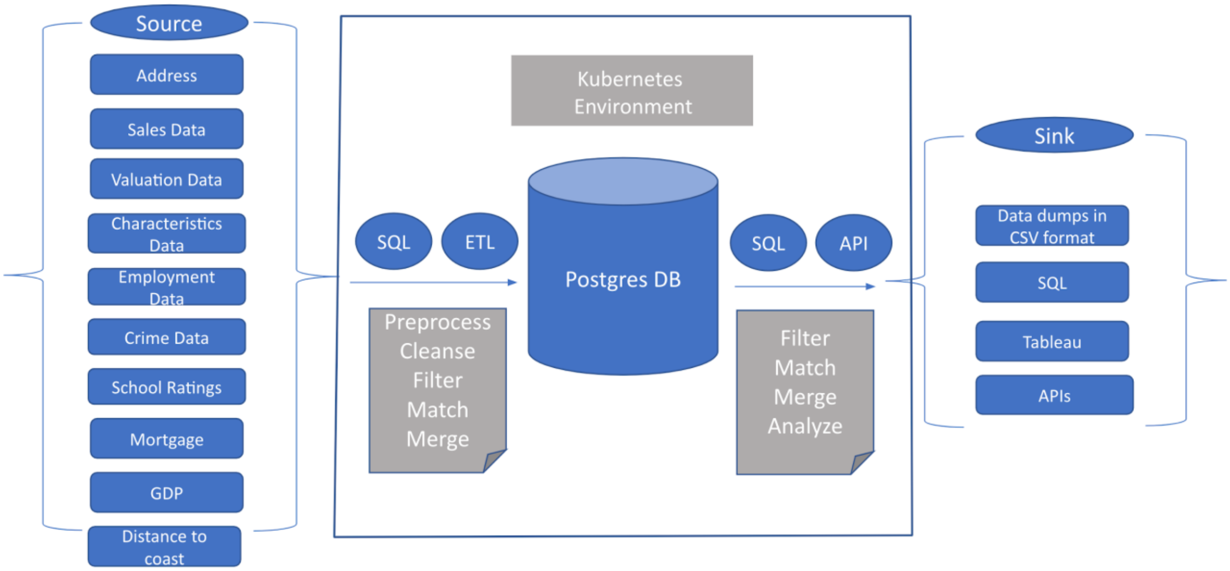


Table: Solution Architecture

Data Deployment

All the raw data (in various formats (CSV, Json)) was loaded in postgres using python and PostgreSQL.
Data is deployed on PostgreSQL server on **kubernetes**.

4. Data Preparation

Data Quality Issues

Outlier Removal :

1. Transaction and property data had properties from outside the San Diego county. These had to be removed.
2. Some properties had very low Sold price - these had to be removed in order for model to work.
3. US Census data was clean
4. Crime data had some zip codes are not related to SD county and had to be removed from the data set.

Data Transformation and Integration for analytics

- Data Cleansing: For invalid content our SQL ingestions and python notebooks altered the data to fit the schema.
- Data Formatting: The schemas were defined in such a way so that they match the incoming data from various sources.
- Data Finding: Primary keys were defined in such a way that they can easily queried and referred as foreign keys from other tables. Example for linking mortgage rates to the sold date was easily done by getting the mortgage rates from the database and linking them to each transaction sold date hence providing the true picture of the macroeconomic mortgage when the house was sold.
- Data Matching: We applied basics string matching, date matching and other factors to link the data sources with each other.
- Data Modeling/Reconstructing: Baseline data was loaded using the provided SQLs. For new data and features we modeled the schema to be as close to the source. While ingesting data we kept what was needed and dropped that was not.

Pre-Processing Methods

Data was loaded directly from the source in various tables. The schema matched the incoming data. Various database views and MAT views were created to query the data that was helpful and needed.

Features Management and Selection

School related features

School related features that were fed into the model included distances (min, average) to nearby schools and the school ratings. After performing deeper analysis of the school related features, we discovered the following issues:

- The distances to schools did not seem correct;
- Serving schools were determined based on the distance. This is typically not how homes are assigned to schools. School districts design school boundaries trying to balance the number and mix of students rather than minimize the distance to the serving school
- In order to minimize the computational burden, the search for serving schools was limited to the same zip code. However, school and school district boundaries have no relation to zip code boundaries.
- Some high schools in San Diego do not have boundaries at all. Students from the entire school district are eligible to attend such schools.

Therefore, we decide to remove the baseline features and add new features dedicated to schools. We realized that adding features related to individual schools may be difficult, due to the same challenges that have been faced before. So we opted for using the rating of the school districts to

which the home belongs, rather than of the individual schools. We used the API available at https://api.schooldigger.com, which provides information about school districts serving a specific location. The location can be specified in terms of lat/lon or address. Here's an example of the url corresponding to a request for school district information for 2202 MR RANCH RD, FALLBROOK
https://api.schooldigger.com/v1.1/districts?appID=2388f8cf&appKey=7da38000f86bcc27e3fa1a10dd92de7e&st=CA&boundaryAddress=2202%20MR%20RANCH%20RD%20FALLBROOK%20CA&isInBoundaryOnly=true
The corresponding response, in JSON format, includes the following information, which we used:

```
"districtList": [
   {
     "districtID": "0613500",
     "districtName": "Fallbrook Union Elementary",
          ...
    "rankHistory": [
       {
         "year": 2018,
         "rank": 225,
         "rankOf": 856,
         "rankStars": 4,
         "rankStatewidePercentage": 73.71,
         "rankScore": 0.6827871
       },...
```

We made over 700,000 calls to this API to retrieve the serving school districts for every property and their rating. The API calls were implemented in Python and timed according to the maximum pace allowed by the API administrator.

Employment related features

We observed that using the raw distance to the employment hubs may not be optimal, because the hubs have different number of employees and average salary. It made sense to reflect that in the distance feature. So we engineered a new feature called proximity in the following way:

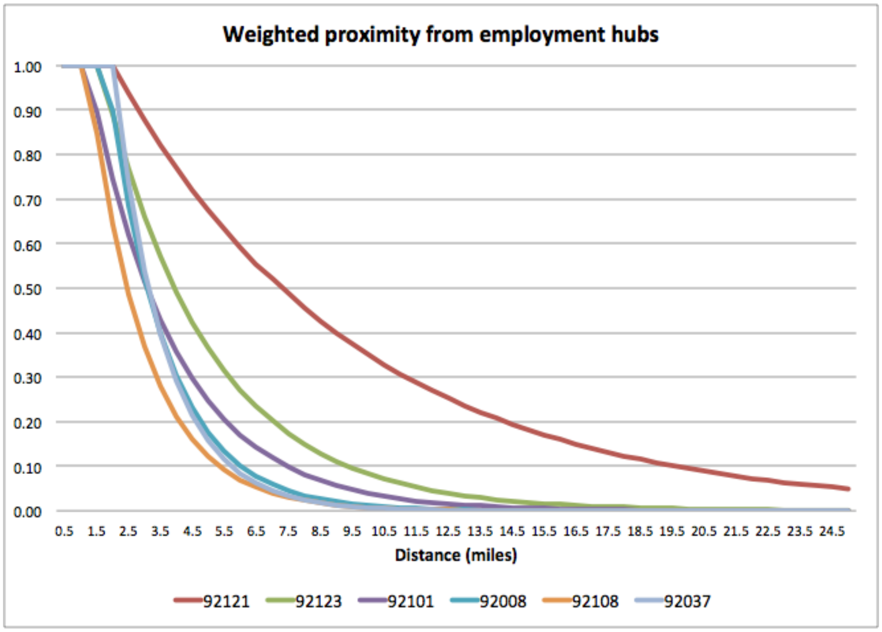
$$\mathit{Proximity}(\mathit{zipcode}_i) = \exp\left(-k\frac{d}{w_i}\right) \in [0,1]$$

d is the distance from the property to zipcode. *k* is a scaling constant, set to 2. The weights *wi* correspond to the % of the total county payroll that was generated in the zipcode, according to the following table:

ZIP	Community	% payroll (*wi*)
92121	San Diego	15.3

92123	San Diego	6.8
92101	San Diego	5.4
92008	Carlsbad	3.7
92108	San Diego	3.6
92037	La Jolla	3.2

The figure below shows the proximity values as functions of the distance for each of the employment hubs.

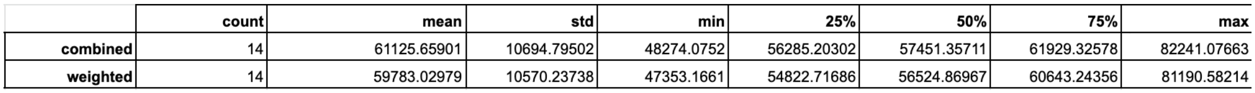


Distance to coast

We used the same approach as for the distances to the employment hubs to convert the distances to proximities. The only difference is that the weight w was set to 1 and the scaling factor *k* was 1.8.

Crime Feature extraction

The initial use case of crime data on the model was just to get the sum of all crime count for all categories per zip code and then divide it by the population of the zip code for a given year. We were not satisfied with the result and crime features didn't show up on the features importance graph. The table below shows the result for just 14 sliding windows from 2012 through 2017.

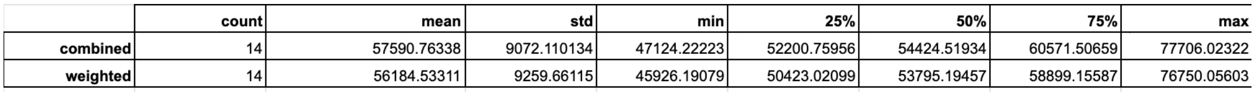


	count	mean	std	min	25%	50%	75%	max
combined	14	61125.65901	10694.79502	48274.0752	56285.20302	57451.35711	61929.32578	82241.07663
weighted	14	59783.02979	10570.23738	47353.1661	54822.71686	56524.86967	60643.24356	81190.58214

So we introduced new engineered features by the below formula.

$$r_{Z,c} = \log(1 + \frac{n_{Z,c}}{\sum_{i=1}^{13} n_{Z,i}})$$

Where $n_{Z,c}$ is the number of crimes of category c in a given zip code Z and i represent 13 different crime categories. We got the best result for Larceny >= $400 category. The table shows the result with the new features. We were able to reduce the Mean RMSE by 6%.



	count	mean	std	min	25%	50%	75%	max
combined	14	57590.76338	9072.110134	47124.22223	52200.75956	54424.51934	60571.50659	77706.02322
weighted	14	56184.53311	9259.66115	45926.19079	50423.02099	53795.19457	58899.15587	76750.05603

5. Analysis Methods

Modeling

Analytic Approach:
We have proposed 4 new factors that determine housing price; particularly we are trying to find out if the suggested features can improve the performance of the base model. Questions below are helping us to define our analytic approach.

1. How does distance from coast affect property value?
2. Does distance to employment hubs affect the fluctuation of the home prices over time?
3. Does mortgage rate as one of our macroeconomic factors that change over the years affect the model?
4. Can crime rate in each zip code help us predict housing price?

We have also introduced **Inflation-Adjusted Sold Price** as new target feature to use in our model.
Inflation adjusted sold price was calculated based on San Diego Home Price Index and Consumer Price Index (CPI). On each iteration, we added one of the introduced features on top of the baseline features and evaluated the performance.

Model Description:
During our model selection process, we started with the baseline features and model that were already given to us. The baseline model is using the **Sliding Window** validation technique. This technique works as the following:

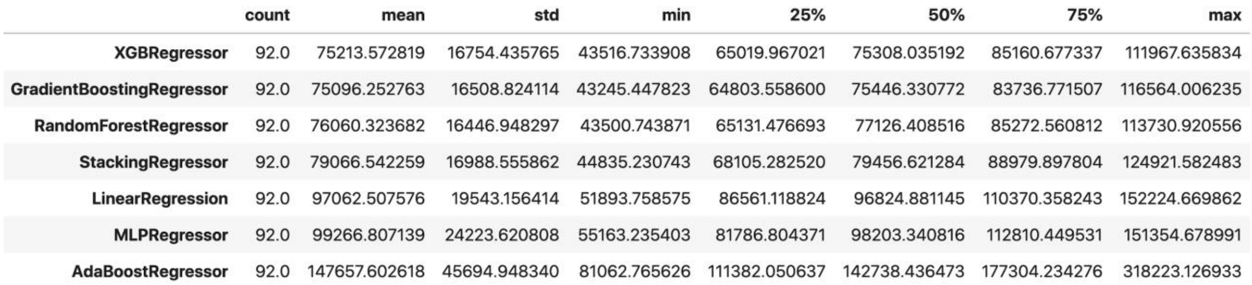
1. Start the window from the earliest date and pick 12 months of data for training. For example, data from 1987-01-01 to 1988-01-01 is used for training.
2. Use the next 4 months of data for testing (i.e. from 1988-01-01 to 1988-05-01). Record the RMSE value, which is used for scoring metric.

3. Slide the beginning of window for 4 months. For example, the beginning of window will move from 1987-01-01 to 1987-05-01.
4. Repeat the steps 1-3 until the end of the sliding window exceeds the latest date in the entire dataset.

With the baseline features, we run 2 baseline regression models (*RandomForestRegressor* and *GradientBoostingRegressor*) and 5 new regressors from *sklearn*, *xgboost*, and *mlxtend* libraries with similar hyperparameters.

Figure 1 and *Table 1* below show the RMSE trend graph and statistical values among those regression models that we tried with the baseline features. *GradientBoostingRegressor* and *XGBRegressor* showed the best results in terms of both **mean** and **median** RMSE scores, which are roughly 75K. Therefore, we decided to continue with the hybrid approach, where the **mean** of RMSE values from both *GradientBoostingRegressor* and *XGBRegressor* models is used as the final evaluation metric.

Figure 1: RMSE trend comparison among various Regression models

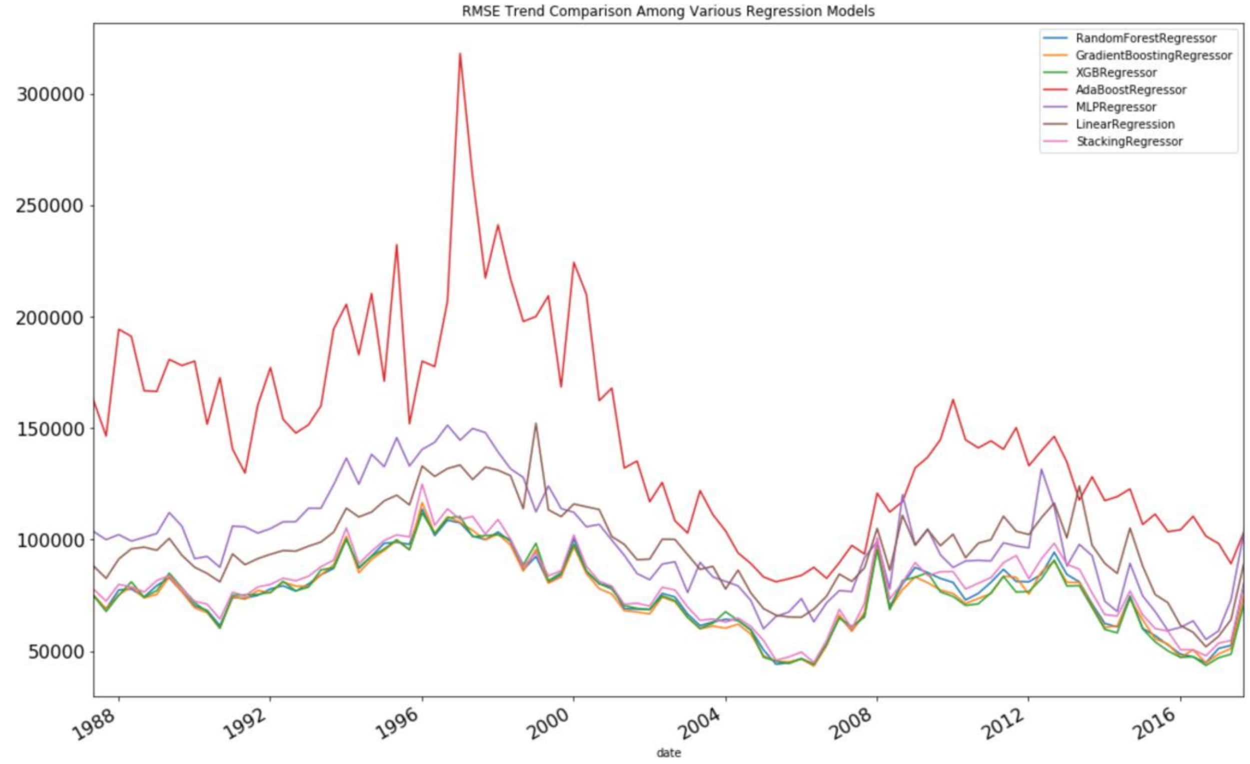


	count	mean	std	min	25%	50%	75%	max
XGBRegressor	92.0	75213.572819	16754.435765	43516.733908	65019.967021	75308.035192	85160.677337	111967.635834
GradientBoostingRegressor	92.0	75096.252763	16508.824114	43245.447823	64803.558600	75446.330772	83736.771507	116564.006235
RandomForestRegressor	92.0	76060.323682	16446.948297	43500.743871	65131.476693	77126.408516	85272.560812	113730.920556
StackingRegressor	92.0	79066.542259	16988.555862	44835.230743	68105.282520	79456.621284	88979.897804	124921.582483
LinearRegression	92.0	97062.507576	19543.156414	51893.758575	86561.118824	96824.881145	110370.358243	152224.669862
MLPRegressor	92.0	99266.807139	24223.620808	55163.235403	81786.804371	98203.340816	112810.449531	151354.678991
AdaBoostRegressor	92.0	147657.602618	45694.948340	81062.765626	111382.050637	142738.436473	177304.234276	318223.126933

Table 1: RMSE statistics (sorted by **median** value) among various Regression models

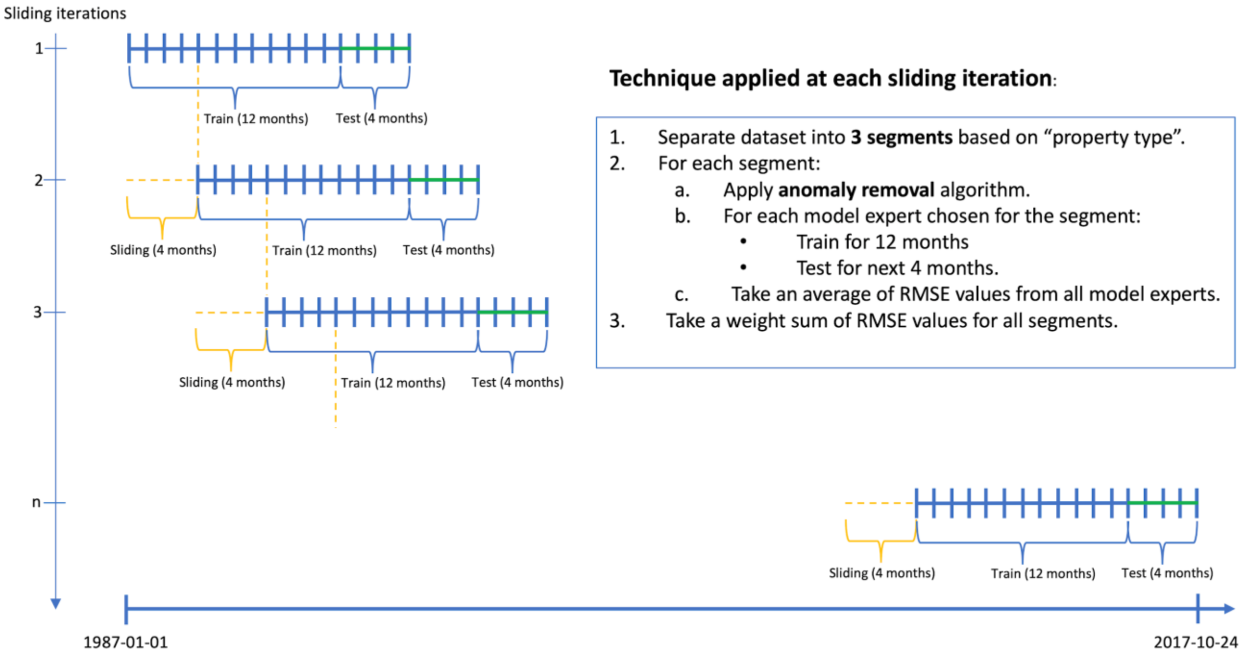
Model Improvement:

To improve our model, we first modified the Sliding Window technique by adding various intermediate steps. The following is a pseudocode for our technique:



```
For each sliding window
    Separate the dataset into **3 segments** based on property type
    For each segment
        Apply **Anomaly Removal** method called **Isolation Forest**
        Apply different **Hybrid** model (GB, XGB) on normal data
    Calculate per-sliding-window RMSE score in 2 different ways
        **Combined:** Combine the predictions from all 3 segments and then
        calculate the RMSE between those predictions and actual values.
        **Weighted:** First calculate the RMSE for individual segments and then
        take the weighted sum of all 3 RMSE values. Weight is proportional to the
        segment’s sample points in each sliding window.
```

Figure 2 below illustrates the modified Sliding Window technique. For 12-month training, 4-month testing, and 4-month sliding window configuration, there are 89 complete iterations for the dataset from 1987-01-01 to 2017-10-24.



Technique applied at each sliding iteration:

1. Separate dataset into **3 segments** based on "property type".
2. For each segment:
   a. Apply **anomaly removal** algorithm.
   b. For each model expert chosen for the segment:
      - Train for 12 months
      - Test for next 4 months.
   c. Take an average of RMSE values from all model experts.
3. Take a weight sum of RMSE values for all segments.

Figure 2: Illustration of the Sliding Window technique

Model Performance:

With the improved Sliding Window technique and newly introduced features, the model performance has improved significantly. *Figure 3* below shows the RMSE trend graph for the evolution of our model.

Blue line represents the baseline model performance (*GradientBoostingRegressor*).
Orange line represents our model with *segmentation* technique.
Green line represents our model with segmentation and *anomaly removal* techniques.
Red line represents our model with segmentation and anomaly removal techniques on top of newly introduced features.

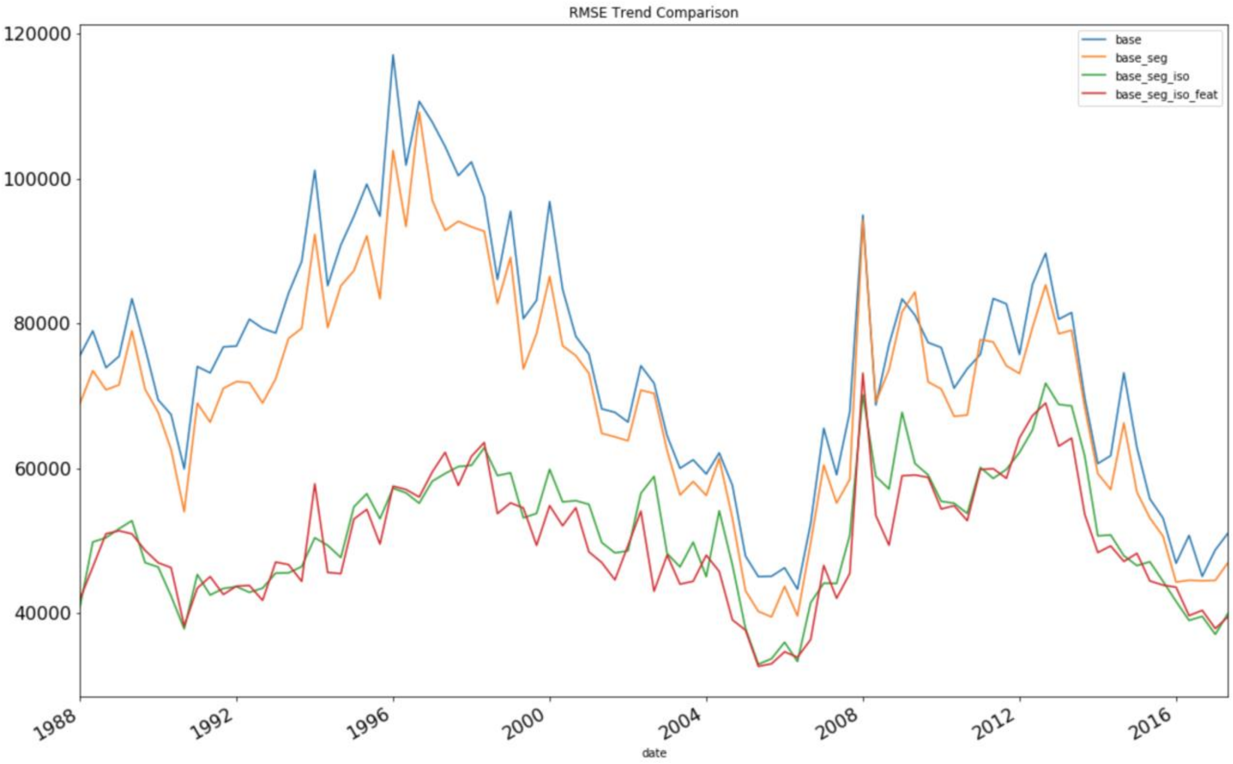
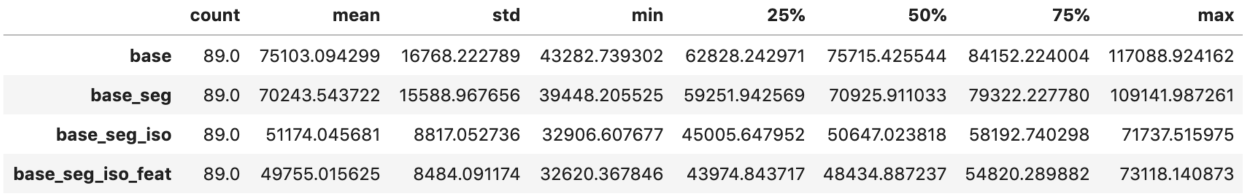


Figure 3: RMSE trend comparison

Looking at the statistics in Table 2 below, we can see that the mean of RMSE score improved at every intermediate step. Our final model show roughly **34% improvement** over the baseline model.



	count	mean	std	min	25%	50%	75%	max
base	89.0	75103.094299	16768.222789	43282.739302	62828.242971	75715.425544	84152.224004	117088.924162
base_seg	89.0	70243.543722	15588.967656	39448.205525	59251.942569	70925.911033	79322.227780	109141.987261
base_seg_iso	89.0	51174.045681	8817.052736	32906.607677	45005.647952	50647.023818	58192.740298	71737.515975
base_seg_iso_feat	89.0	49755.015625	8484.091174	32620.367846	43974.843717	48434.887237	54820.289882	73118.140873

Table 2: RMSE statistics for the baseline and improved models

6. Findings and Reporting

Key findings

By looking at the RMSE trendline in Figure 3, we observe that each additional refinement step of our model resulted in the RMSE reduction. The largest portion of the improvement is attributed to the removal of outliers using the isolation forest model. This testifies to the fact that careful data preparation, preprocessing and outlier removal is a key element of a successful model.

We further observe that simple adjustment of the model to the dataset, namely: segmentation of the dataset based on the property type (condo, SF, MF), can yield substantial improvement.

Next, we observe that our feature engineering provided visible improvement, albeit not as much as the other refinements we made.

Finally, we observe that the model performance varies substantially over the time period between 1988 and 2017. In particular, the RMSE increases between 1996 and 2004 and then again between 2009 and 2012. The former time period corresponds to the housing market that ended in the housing bubble burst in 2006. In this time period, there was a significant amount of "irrationality" in the market, reflected in the readiness of buyer to pay prices above the market value. The latter time period corresponds to the recession, when buyers or banks were selling properties below the market value, again exhibiting "irrational" behavior. We suspect that this "irrationality" in the market, which cannot be modelled by the features that we used in our model, is the main reason for the spike in the RMSE during these two periods. Looking at the raw % error trend in Figure 4, we observe that, indeed, the model has difficulties tracking the market between 1998 and 2006, which is reflected in the rising amplitudes of the error spikes. The model attempts to understand the market, using the moving average/sliding window mechanism, which is reflected in frequent zero crossing of the error, but the "irrationality" in the market escapes it. Similar observations can be made for the time between 2008 and 2012.

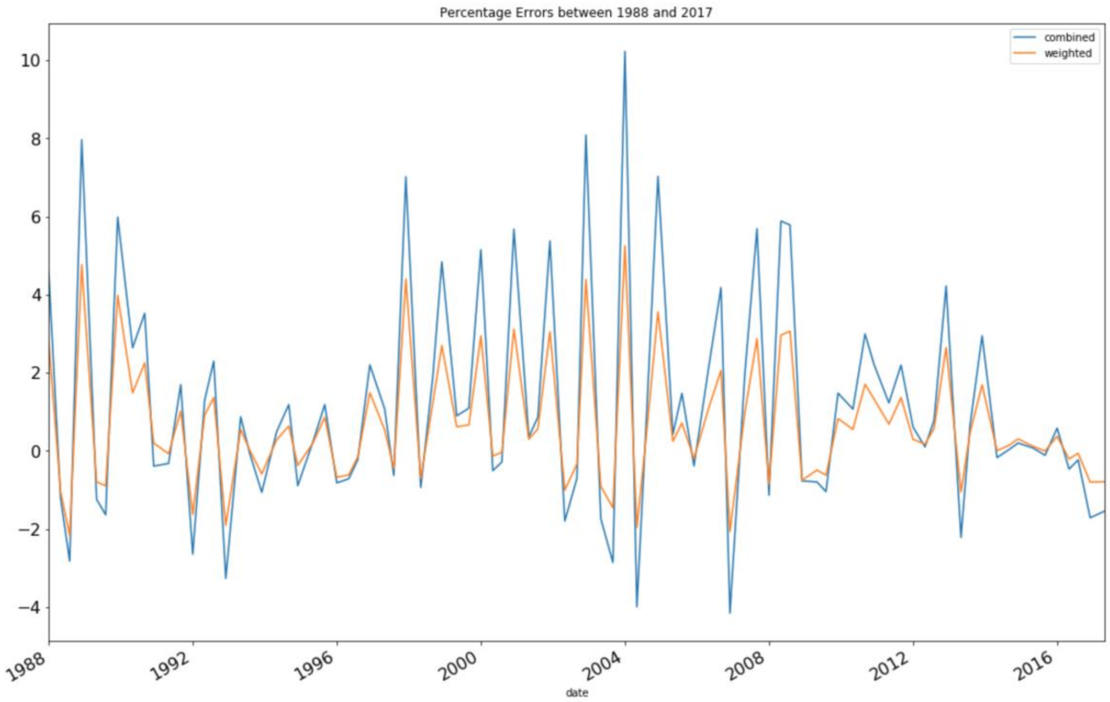


Figure 3: raw % error trend

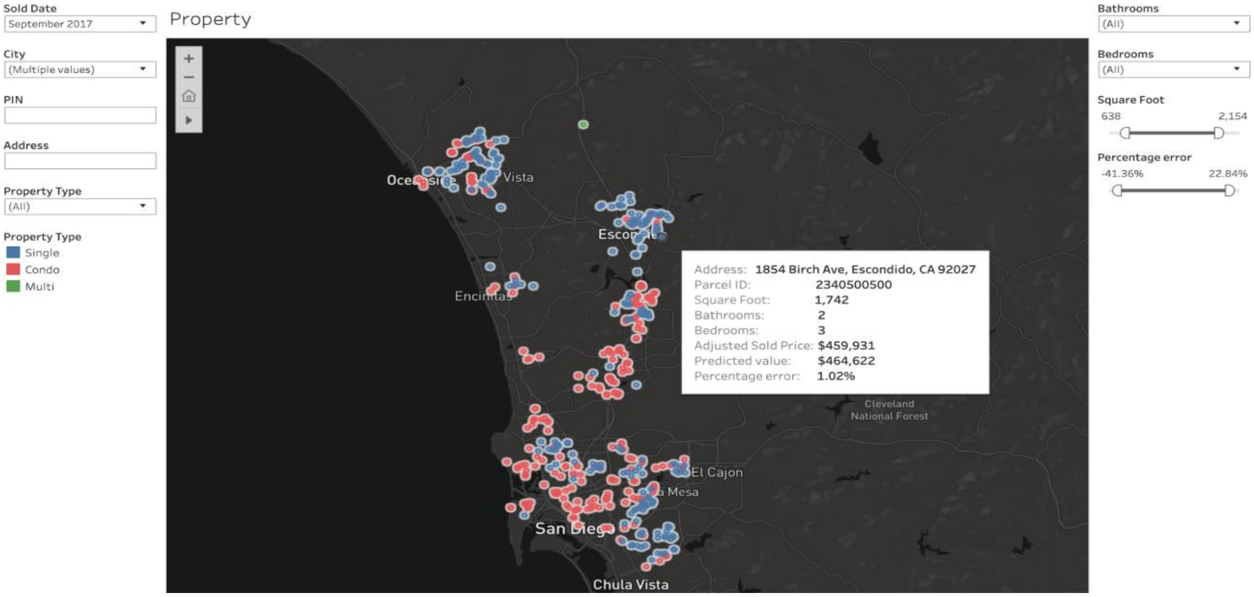
Reporting and presenting the findings

The RMSE is the key performance measure for our model. So we focused on the presenting the RMSE and its trends. We made sure that we exhibit the impact of the different improvement steps we undertook.

It is also important to demonstrate the performance of our model on specific properties and compare it to the commercial tools, such as Zillow. We developed two dashboards to showcase how our model works on a random property and how it performs compared to Zillow.

Property Type	Parcel ID	Address	Zip Code	Adjusted Sold Price/ Date	DSE C4 Predicted Value	Adjusted Zillow Estimate	% error	% error
Single Family	5772921800	8131 Brennan St.	92114	$432,335 Sep, 2017	$428,471	$454,207	-0.89%	5.06%
Single Family	1694914100	4911 Kalamis Way	92056	$517,928 Nov, 2009	$547,802	$487,751	5.77%	-5.82%
Condo	3451420803	4435 Nobel Dr Unit #3	92122	$316,842 Sep, 2017	$305,314	$322,463	-3.64%	1.77%

Multi	5490732900	635 Stork St.	92114	$419,503 Nov, 2016	$386,121	$420,565	-7.96%	0.25%



Sold Date: September 2017

City: (Multiple values)

PIN:

Address:

Property Type: (All)

Property Type
- 🟦 Single
- 🟥 Condo
- 🟩 Multi

Property

Bathrooms: (All)

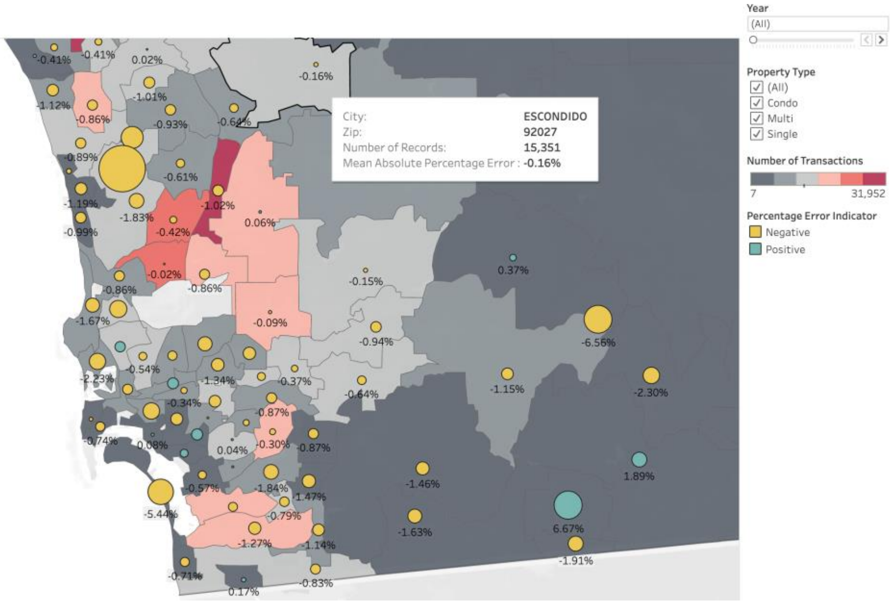
Bedrooms: (All)

Square Foot: 638 – 2,154

Percentage error: -41.36% – 22.84%

Address: 1854 Birch Ave, Escondido, CA 92027
Parcel ID: 2340500500
Square Foot: 1,742
Bathrooms: 2
Bedrooms: 3
Adjusted Sold Price: $459,931
Predicted value: $464,622
Percentage error: 1.02%

The dashboard below shows the number of transaction on a given year (background color) while the radius of circles represents the Mean absolute percentage error for each zip code. The color shows the percentage error indicator as well.



City: ESCONDIDO
Zip: 92027
Number of Records: 15,351
Mean Absolute Percentage Error : -0.16%

7. Solution Architecture, Performance and Evaluation

Performance measure

The key performance measure was the RMSE. We were able to reduce the RMSE w.r.t. The baseline by 34%. Additional performance measures of interest are:

- Standard deviation of the RMSE, which was reduced by ~50% compared to the baseline
- Maximum RMSE, which was reduced by ~40% compared to the baseline

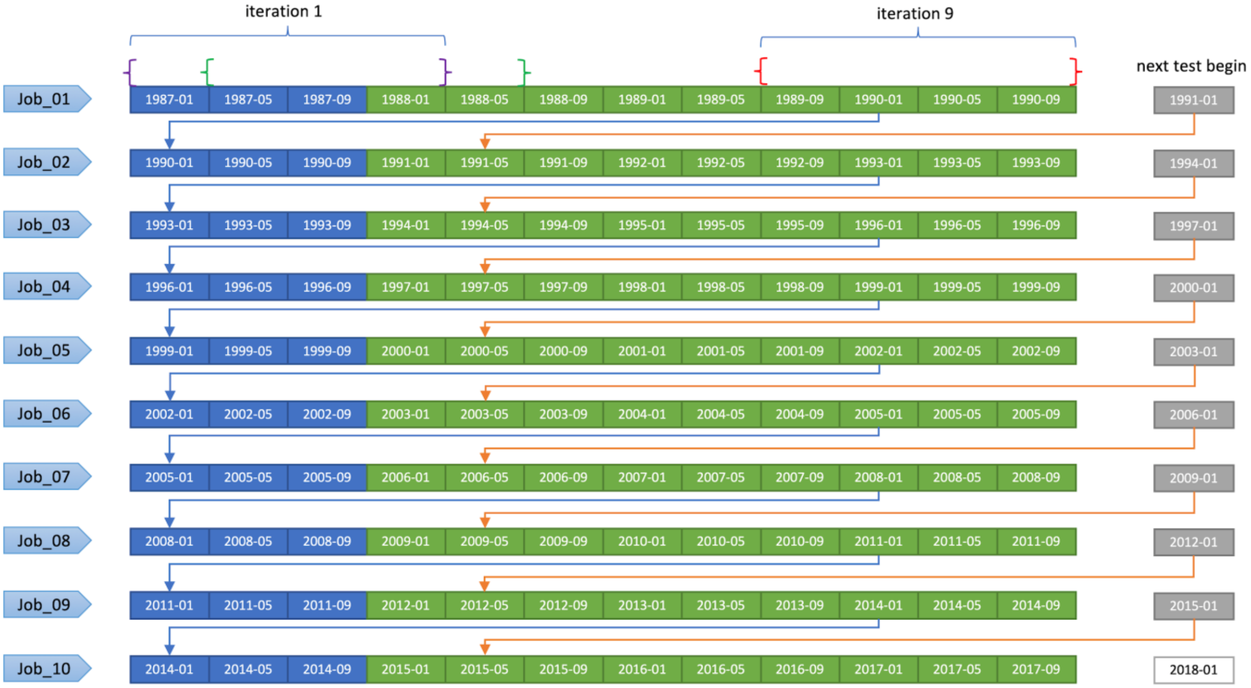
In addition to the improved performance in terms of RMSE, we also improved the performance in terms of the time needed to run and tune the model, as described in the subsequent sections.

Scalability

As usual in any Machine Learning project, the hyperparameter tuning process was the most compute intensive stage for this project. We wanted to try 1632 different configurations of hyperparameters for each of 3 segments and for each of 90 sliding windows. That would be 1632 * 3 * 90 = **440,640 unit time**. Let’s assume that the function calls (*fit*, *predict*, *mean_square_error*, *sqrt*) take only 2 minutes on average for each segment given the size of our dataset. If we were to run the tuning on a single core, it would take 440,640 * 2 = 881,280 minutes = 612 days = **1 year 8 months 1 week.** Since we didn’t have that much time for this project, we had to find an alternative solution for this problem. Our solution was **data** and **task** parallelism.

Data Parallelism:
We split the entire dataset into 10 chunks. Each chunk contains data for 48 months. There is a 12-month overlap between consecutive chunks because first 12-month for each chunk will be the starting data for training.

Figure 4 below shows how we split the entire dataset for a massively parallel hyperparameter tuning.

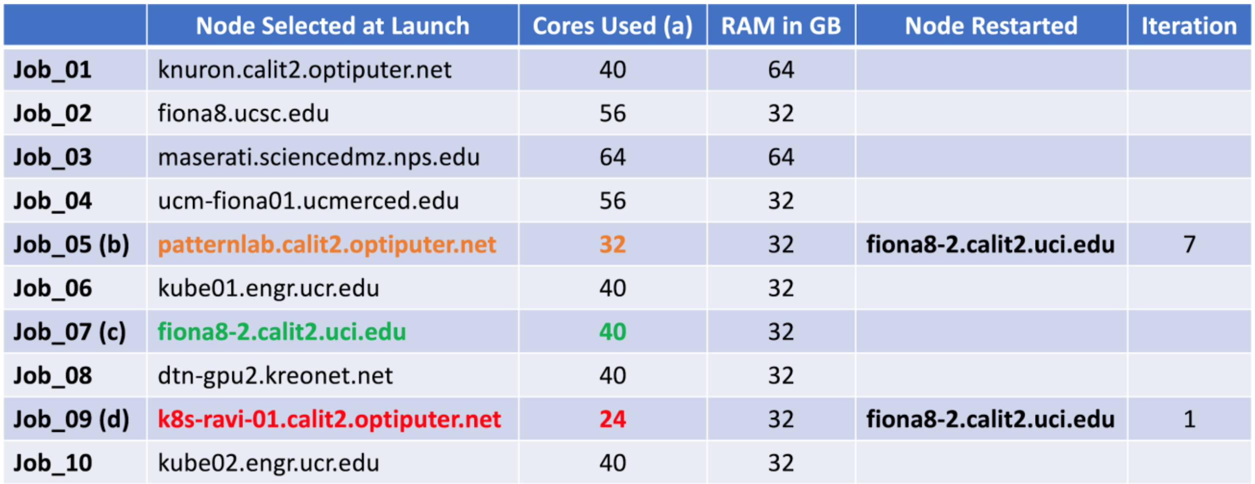


													next test begin
Job_01	1987-01	1987-05	1987-09	1988-01	1988-05	1988-09	1989-01	1989-05	1989-09	1990-01	1990-05	1990-09	1991-01
Job_02	1990-01	1990-05	1990-09	1991-01	1991-05	1991-09	1992-01	1992-05	1992-09	1993-01	1993-05	1993-09	1994-01
Job_03	1993-01	1993-05	1993-09	1994-01	1994-05	1994-09	1995-01	1995-05	1995-09	1996-01	1996-05	1996-09	1997-01
Job_04	1996-01	1996-05	1996-09	1997-01	1997-05	1997-09	1998-01	1998-05	1998-09	1999-01	1999-05	1999-09	2000-01
Job_05	1999-01	1999-05	1999-09	2000-01	2000-05	2000-09	2001-01	2001-05	2001-09	2002-01	2002-05	2002-09	2003-01
Job_06	2002-01	2002-05	2002-09	2003-01	2003-05	2003-09	2004-01	2004-05	2004-09	2005-01	2005-05	2005-09	2006-01
Job_07	2005-01	2005-05	2005-09	2006-01	2006-05	2006-09	2007-01	2007-05	2007-09	2008-01	2008-05	2008-09	2009-01
Job_08	2008-01	2008-05	2008-09	2009-01	2009-05	2009-09	2010-01	2010-05	2010-09	2011-01	2011-05	2011-09	2012-01
Job_09	2011-01	2011-05	2011-09	2012-01	2012-05	2012-09	2013-01	2013-05	2013-09	2014-01	2014-05	2014-09	2015-01
Job_10	2014-01	2014-05	2014-09	2015-01	2015-05	2015-09	2016-01	2016-05	2016-09	2017-01	2017-05	2017-09	2018-01

Figure 4: Dataset splitting for a massively parallel hyperparameter tuning

Task Parallelism:
Each data chunk was offloaded with a separate “Job” to different nodes in the Kubernetes cluster at the same time. Kubernetes nodes were carefully selected among those that have higher number of CPU cores so that we could utilize the most of them. For example, if a particular node has 48 cores, there will be 48 different models’ task that runs on exactly the same subset of data on a given sliding window.

Table 3 below shows the names of Kubernetes node that we selected for each job. All except two nodes had 48 or more cores. At each node, we used *CoreCount - 8* cores. In other words, if a particular node had 64 cores, we used 54 of them (64 - 8).



	Node Selected at Launch	Cores Used (a)	RAM in GB	Node Restarted	Iteration
Job_01	knuron.calit2.optiputer.net	40	64		
Job_02	fiona8.ucsc.edu	56	32		
Job_03	maserati.sciencedmz.nps.edu	64	64		
Job_04	ucm-fiona01.ucmerced.edu	56	32		
Job_05 (b)	**patternlab.calit2.optiputer.net**	**32**	32	**fiona8-2.calit2.uci.edu**	7
Job_06	kube01.engr.ucr.edu	40	32		
Job_07 (c)	**fiona8-2.calit2.uci.edu**	**40**	32		
Job_08	dtn-gpu2.kreonet.net	40	32		
Job_09 (d)	**k8s-ravi-01.calit2.optiputer.net**	**24**	32	**fiona8-2.calit2.uci.edu**	1
Job_10	kube02.engr.ucr.edu	40	32		

Table 3: Kubernetes nodes selected for a massively parallel hyperparameter tuning

Hyperparameter tuning completion:

As shown in *Table 4* below, with the massively parallel computation, entire hyperparameter tuning process took only 4546 minutes = 3 days 4 hours. That was only 0.5% of our initial estimate. In other words, with the data and task parallelism approach, we increased the hyperparameter tuning process by **200 times** in speed.

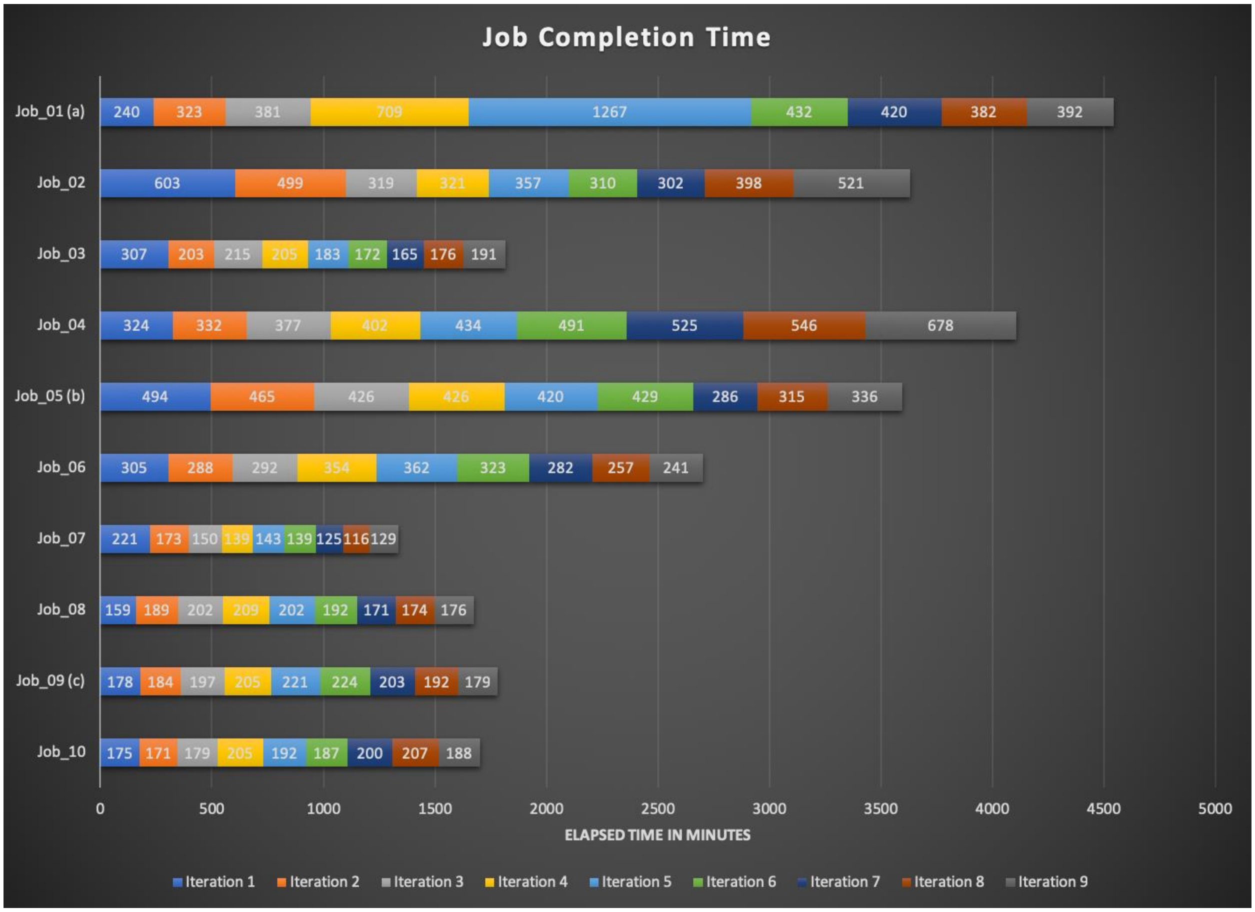


Table 3: Kubernetes jobs completion time

Scalability solution:

If we were to scale this project to State (aka. California) and maybe to Country level, we could solve the Big Data problem with the similar data and task parallelism approach. Dataset will vary significantly from state to state. Therefore, each state may require different model. That will make data parallelism easier. As long as we have enough resources in Kubernetes clusters, task parallelism shouldn't be a problem.

Budget management

Since we did not use Amazon Web Services on our project, Budget management was not our concern. However, we spent 10% of our budget for SchoolDigger API and also updating latitude and longitude features on the dataset.

8. Conclusions:

The initial goal was to get a better RMSE (10% or better) than the baseline work, however we were able to improve RMSE by 34%. Based on our observation, during normal market (1994-2000) and (2013 -2017), the percentage error is almost zero, but during the bubble and crash markets, the error spikes up to 5.25% on the weighted version. Basically, the model attempts to converge on the changes in the market gradually. All in all, we may have reached a ceiling by improving the model and adding more features.

Appendices

A. DSE MAS Knowledge Applied to the Project

Almost every skill learnt as part of the program was utilized in designing and implementing the project. Python for Data Analysis was key as all our code and analysis used Python and the associated modules and packages for database(sqlalchemy), file loading and parsing (numpy, pandas) and scikit-learn were used heavily. Matplotlib was used to create plots and graphs. Postgres was used as the database management system. We used Materialized views heavily to materialize the data so our queries could run in an optimized manner. Once data was loaded, ETL was performed and clean data was ingested in other tables which were later referred by the MAT views and used for EDA and Analysis. Tableau and Matplotlib were key to do all analysis and really helped with feature engineering. Key evaluation metrics like RMSE , Standard deviation, mean absolute error were used as a comparison with baseline work to determine the improvement our work achieved. Various machine learning algorithms were used to model the data. XGB, GB, RandomForest to name a few. Others like Adaboost were experimented with but did not perform well on our data. Lastly, for visualization Tableau was key to create a real life demonstration of predicted values of the properties that our model generated. Overall, every course in the curriculum was essential to get the project done.

B. Data and Software Archive for Reproducibility

Project Links

1. **Digital Object Identifier (DOI):** https://doi.org/10.6075/J0891459.
   Dataset: https://drive.google.com/open?id=1Pbtbdz_P3eFnIUstv4AjZuE1bArbmZ0g
   Main Google drive: https://drive.google.com/open?id=1IKc6LpmMBrBAQe00BWa3CbPSovx_MY5e
   Output Result: /cephfs/housing/mas19/ on https://housing-jupyter.nautilus.optiputer.net
   Dataset description: https://github.com/alexyanw/dse_capstone/tree/master/data

2. **Report, notebooks and source code.**

All source code and notebooks are uploaded to github project, the project folder is well structured, refer to README: https://github.com/vvural/Housing/blob/master/mas19/readme.txt
Python notebooks: https://github.com/vvural/Housing/blob/master/mas19/runner.ipynb
Python source code: https://github.com/vvural/Housing/tree/master/mas19/src
Report and presentation: https://drive.google.com/open?id=1pXlVXnfCCa4UIAqoH3x-D0CsUlf_CXzO

Tools Set Up

1. **PostgresSQL**

PostgreSQL 9.6 and below extensions installed

```
CREATE EXTENSION postgis;
CREATE EXTENSION fuzzystrmatch; --needed for postgis_tiger_geocoder
CREATE EXTENSION address_standardizer;
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION postgis_tiger_geocoder;
```

2. Tableau Workbook

Tableau workbook is located at the path below named “V1.twb”; all datasource needed are saved in the same location in a csv file called “df_pred_new_lat_lon.csv”

https://drive.google.com/open?id=1XVzun13-4437xd8ceW65HxFuOCQ7f0Fc