MAT 494
Arizona Real Estate Analysis

Dean Brasen, Joshua Tenorio

Arizona State University

November 2022

Introduction

- Goal: taking real estate Arizona data to make models and predictions about house prices
- Utilizes various data science methods and implementations in Python

Data Description

- Kaggle dataset "Arizona Houses 2021" was used [N], which included 563 houses (of which, 562 was used).
- Multiple inputs from this dataset were used as independent variables:
  - Zipcode
  - number of beds and baths
  - square footage
- We created additional features to better categorize the dataset:
  - region index (North/Central/South AZ)
  - city index (represents how expensive living in a given city is)

Math Model Description

- Models used:
  - K-Means
  - Gradient Boosting

Cluster Analysis: Motivation

- Extremely useful to group similar variables under a single point, or a centroid.
- Allows real estate data to be grouped in an easier way to model prices based on given variables

K-Means Algorithm

- Goal: Cluster data points around K given centroids [Mora]
- Defined by an algorithm that determines relative distances to randomly assigned "centroids" then reshifts the centroid point that maximizes distance
- Silhouette scores are used to determine the best K that minimizes the overall distances from all given data points to K centroids
- K means algorithm is repeatedly computed in the event that the wrong centroids are calibrated as the true points.

Implementing K-Means Algorithm

The code below utilizes the Scipy library which already has a Kmeans function included.

```
import pandas as pd
from scipy.cluster.vq import vq, kmeans, whiten
import numpy as np

gps = pd.read_csv("data/interim/with-gps.csv")
coords = []
for idx, row in gps.iterrows():
    lat = row["latitude"]
    lon = row["longitude"]
    pair = []
    pair.append(lon) # x axis
    pair.append(lat) # y axis
    coords.append(pair)

coords_np = np.array(coords)

# Whiten (i.e. normalize) data
whitened = whiten(coords_np)

# Find 3 clusters in the data
codebook, distort = kmeans(whitened, 3)
# codebook is list of cluster centroids, distort is additional data
```

Gradient Boosting

- Goal: minimize a loss function using an iterative process of "weak learners" [Morb]
- Although there are multiple variables that can affect the learning process, hyperparamter tuning can be used to optimize the algorithm
- typically a faster learning rate with more trees is preferrable when utilizing gradient boosting
- Extremely useful in the field of machine learning

Implementing Gradient Boosting

The code below displays an algorithm to implement Gradient Boosting on the negative mean squared error loss function.

```
def gb_score(X, y):
    # Model assignment
    model = GradientBoostingRegressor
    # Perform Grid-Search
    gsc = GridSearchCV(
        estimator=model(),
        param_grid={
            'learning_rate': [0.01, 0.1, 0.2, 0.3],
            'n_estimators': [100, 300, 500, 1000],
        },
        cv=5,
        scoring='neg_mean_squared_error',
        verbose=0,
        n_jobs=-1)
    # Finding the best parameter
    grid_result = gsc.fit(X, y)
    best_params = grid_result.best_params_
    # Random forest regressor with specification
    gb = model(learning_rate=best_params["learning_rate"],
               n_estimators=best_params["n_estimators"],
               random_state=False,
               verbose=False)
    # Apply cross validiation on the model
    gb.fit(X, y)
    score = cv_score(gb, X, y, 5)
    # Return information
    return score, best_params, gb
```

```
def cv_score(model, X, y, groups):
    # Perform the cross validation
    scores = cross_val_score(model, X, y,
                             cv=groups,
                             scoring='neg_mean_squared_error')
    # Taking the expe
    # of the numbers we are getting
    corrected_score = [np.sqrt(-x) for x in scores]
    return corrected_score
```

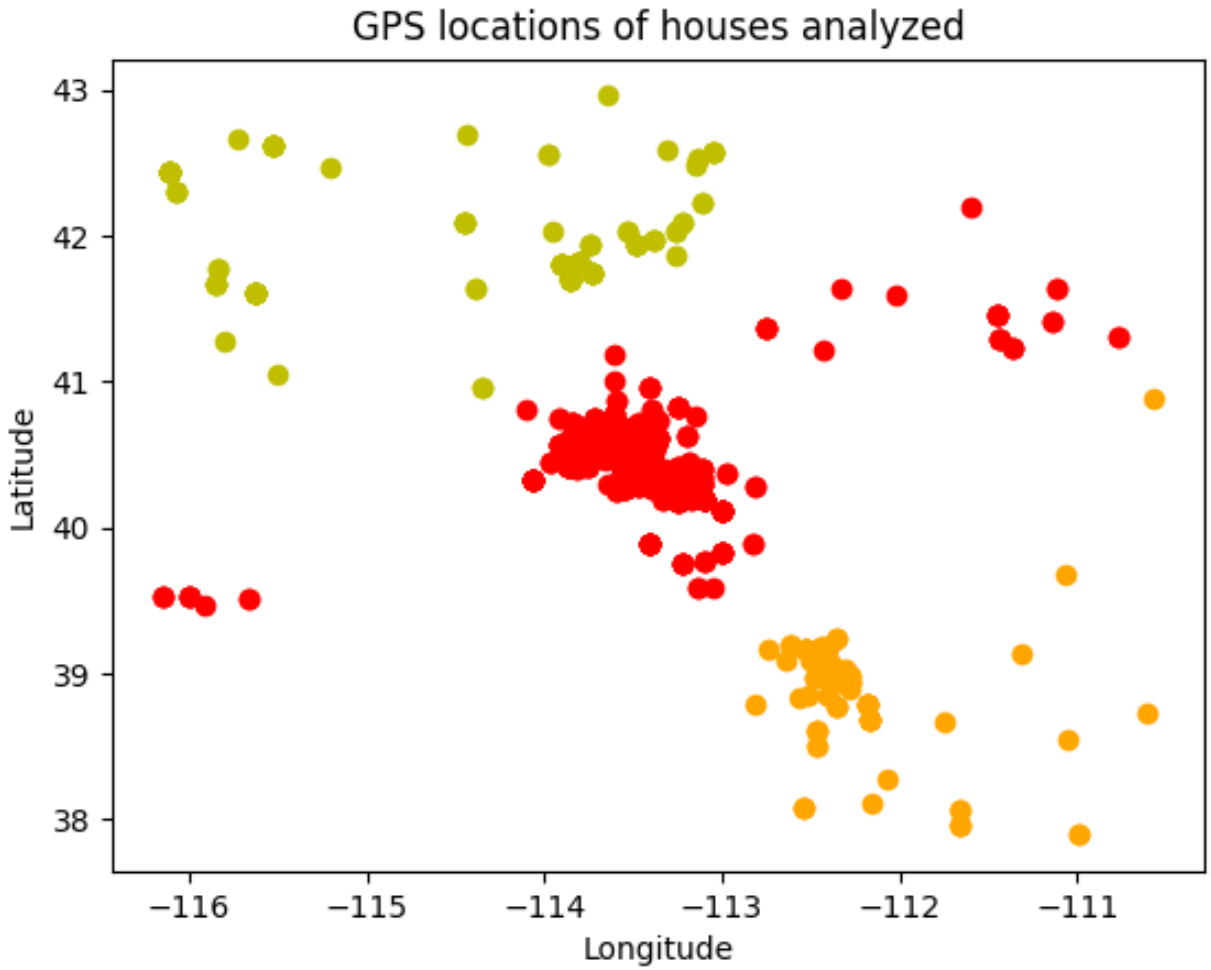
Model Prediction

Our Model prediction was broken up into three distinct steps:

1. Assign houses into regions
2. Assign houses a city index
3. Utilise Gradient Boosting to train the model and make a prediction

Model Prediction: Regions

To divide the dataset into three distinct regions (i.e. North/Central/South AZ) K-means clustering was used. The results of this step is seen below:



Model Prediction: City Index

We wanted to assign each house a city index. City indices were determined by how expensive a given city was to live in, where the higher the index the more expensive a city is.
The code for this can be seen on the next slide.

Model Prediction: City Index

```python
import pandas as pd

# order cities by how expensive they are
raw = pd.read_csv("data/processed/cleaned-final.csv")
grouped = raw.groupby('Local_area').sum().reset_index()
total = grouped.sort_values('Price', ascending=False)

cities = []
for idx, row in total.iterrows():
    if row["Local_area"] not in cities:
        cities.append(row["Local_area"])

# assign cities their indices
city_idx = []
for i in range(len(raw)):
    city_idx.append(0)

def get_city_idx(name):
    i = 0
    for city in cities:
        if name == city:
            return i
        else:
            i += 1
    return i

# assign city indices to each row
raw["city_index"] = city_idx
for idx, row in raw.iterrows():
    raw.at[idx, "city_index"] = get_city_idx(row["Local_area"])

# save data
raw.to_csv("data/processed/final-final.csv")
```

Model Prediction: Gradient Boosting

Gradient boosting was then used to train the model.

- In order to select the optimal number of trees and the learning rate, GridSearchCV was used.

The model was trained on 85% of the dataset, meaning 15% would be used for validation of the model.

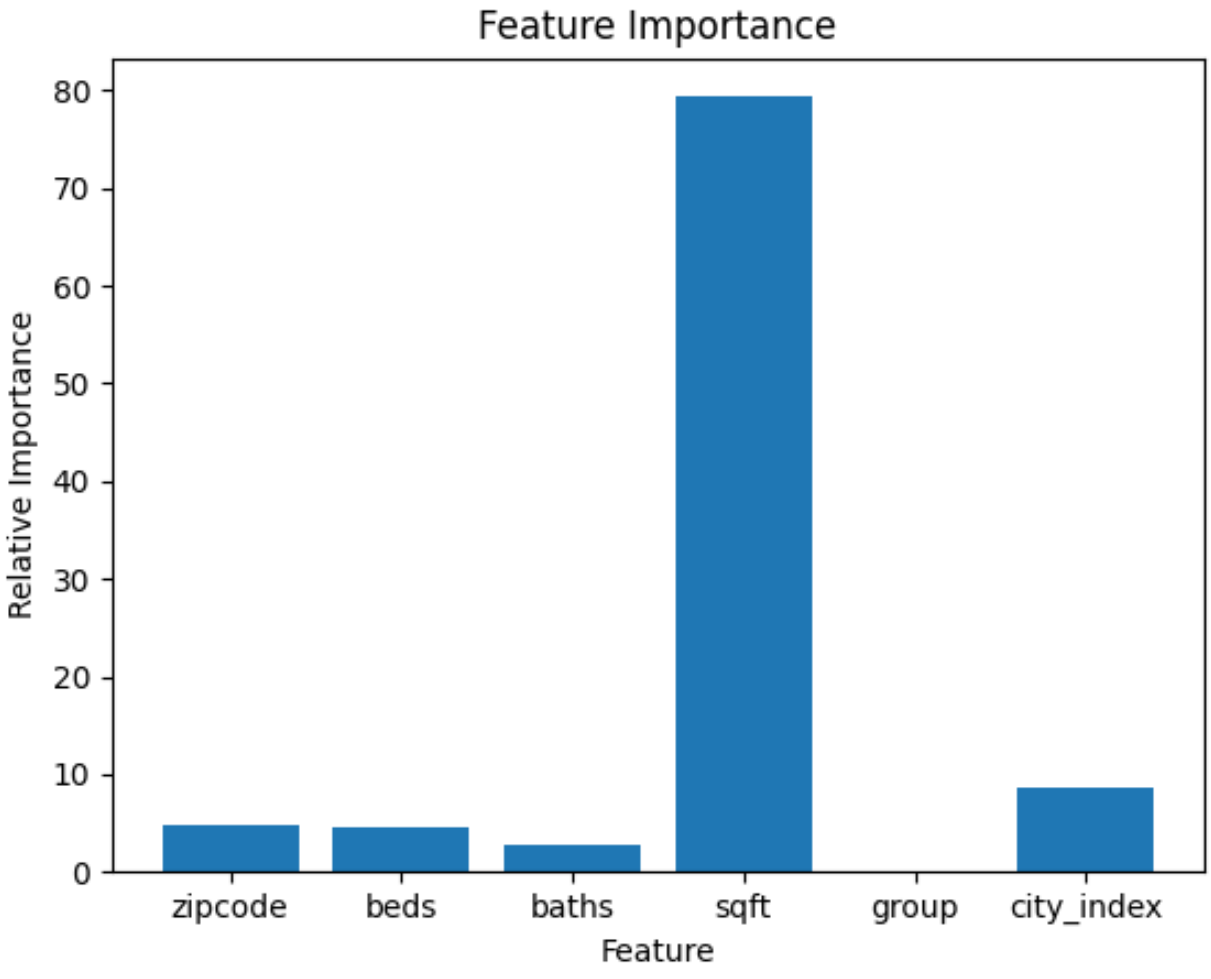
Discussion

The absolute percent error between the predicted price and actual price of the houses in the test dataset was used to validate the model, the results of which are seen below.

Mean	Std. Dev	Minimum	1st Quartile	Median	3rd Quartile	
62.870	155.192	0.420	12.739	23.304	41.344	

Discussion

This chart shows the feature importance.



Discussion

One way the results could've been more accurate was by increasing the amount of test data. We were using 562 out of the 563 rows from the original data set, and of those 562 only 85% was used to train the model. This could be accomplished by scraping Zillow for more houses. Although we would need to clean up the data, this would allow us to create a more accurate prediction model.

Another way to obtain more accurate results would be to focus on a specific region. We conducted analysis on all parts of the state, but focusing on a specific area (such as Maricopa county) would reduce variance in the data.

References

[Mora] Paul Mora. *End-To-End Machine Learning Project — 04 Clustering*. URL: https://www.data4help.org/blog/end-to-end-machine-learning-project-04-clustering. (accessed: October 15 2022).

[Morb] Paul Mora. *End-To-End Machine Learning Project — 06 Predicting Real Estate Prices with Machine Learning*. URL: https://www.data4help.org/blog/end-to-end-machine-learning-project-06-predicting-real-estate-prices-with-machine-learning. (accessed: October 15 2022).

[N] Antonio N. *Arizona Houses 2021*. URL: https://www.kaggle.com/datasets/antoniong203/arizona-houses-2021. Version 1, (accessed: October 15 2022).