Machine Learning: Real Estate Prediction Model

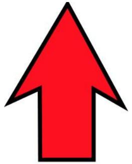
Joe Awanis
Manitha Ayaninilkkunnathil
Jean Paul Nduwayo Ntore
Cory Peacock

https://github.com/josephawanis30/MachineLearningRealEstatePredictions

Overview

The median sale price of a home in central Phoenix:

$425,000



27.6%

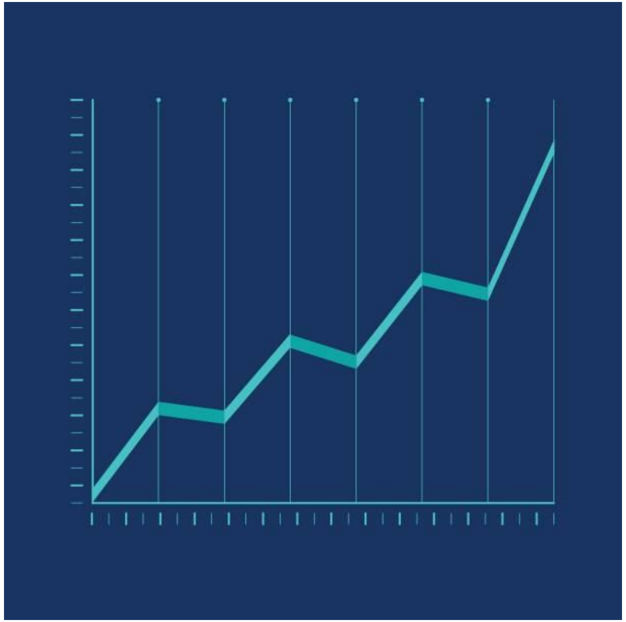
since May 5, 2021

Overview

There are, of course, numerous features that factor into the listing price for any realty listing. On one end of the spectrum, **large data-driven companies like Zillow are able to utilize their algorithms to price homes very accurately**. According to Zillow's website,2 their nationwide median error rate for an on-market home "Zestimate" is 1.9%. On the other end of the spectrum, **independent realtors utilize a mix of experiential intuition and "comps"** - hyper local comparable house listings. According to Brian Houle, a local Phoenix Realtor, finding six to ten homes within a two-mile radius with a similar square footage and similar housing condition / quality is a good starting place for finding accurate pricing. What independent Realtors have internalized as intuition, data-driven Realtors have made explicit in their algorithms, the features of which are typically not publicly available.

Overview

The primary purpose of this analysis is to create a machine learning model that can predict housing prices for single-family detached homes in the central Phoenix area utilizing features that are publicly available.



Data Sourcing: FlexMLS Realty Portal

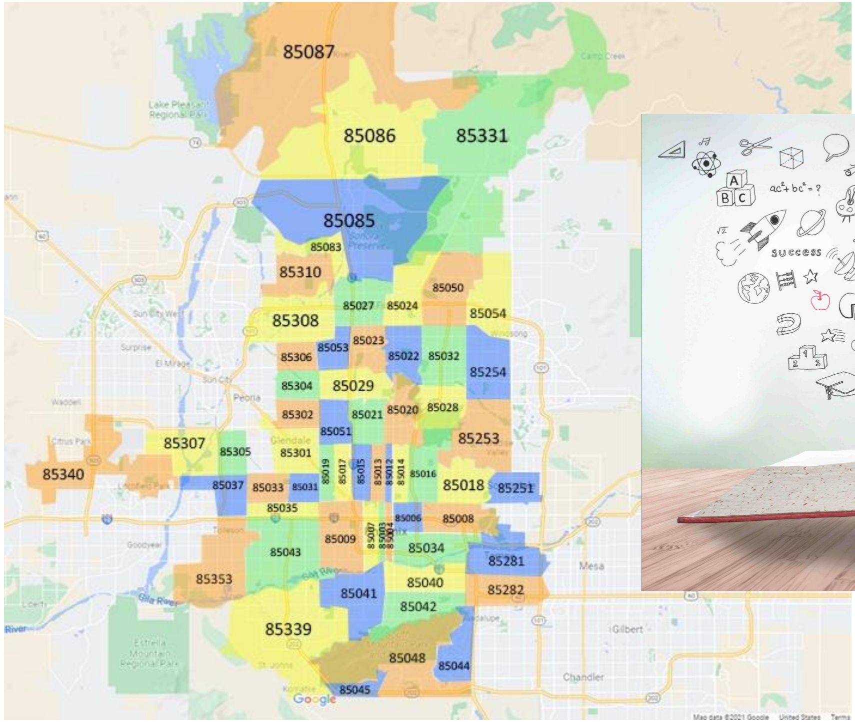
- Search conducted on May 6 and 7, 2022
  - Central Phoenix
  - Asking price of $1,300,000 or less
- More than 2,300 houses for sale or recently sold

Data Sourcing: FlexMLS Realty Portal

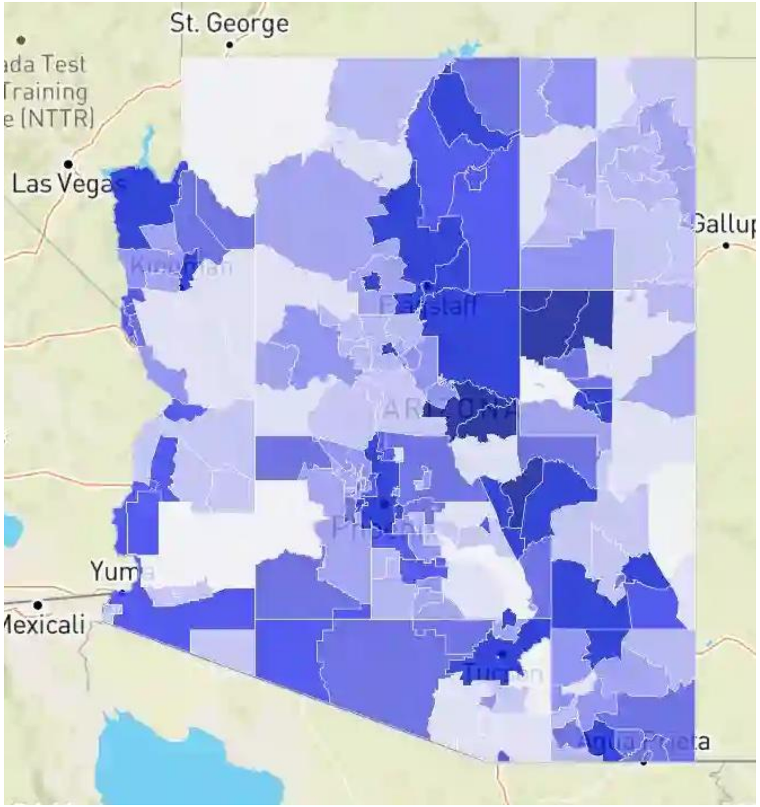
The features of each data point include:

- Price (Asking Price / Sale Price)
- Zip
- Year built
- Bedrooms
- Bathrooms
- Approximate Square Footage

Data Sourcing: Arizona Department of Education



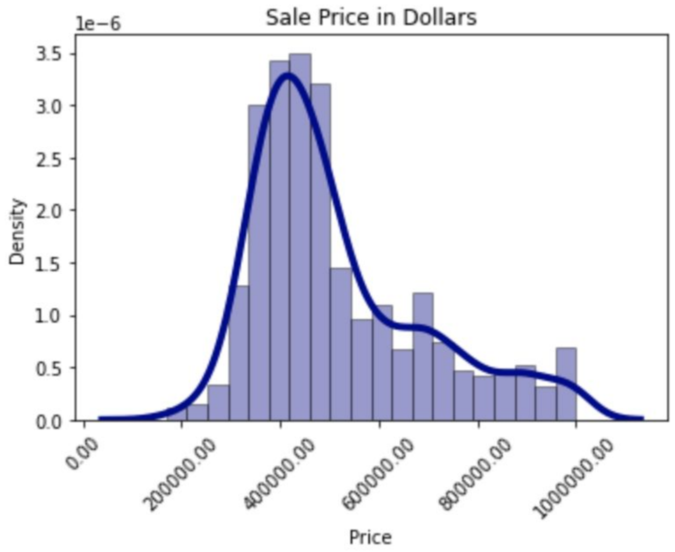
Data Sourcing: Phoenix.gov Crime



Data Exploration: Descriptive Statistics

A density plot for price:

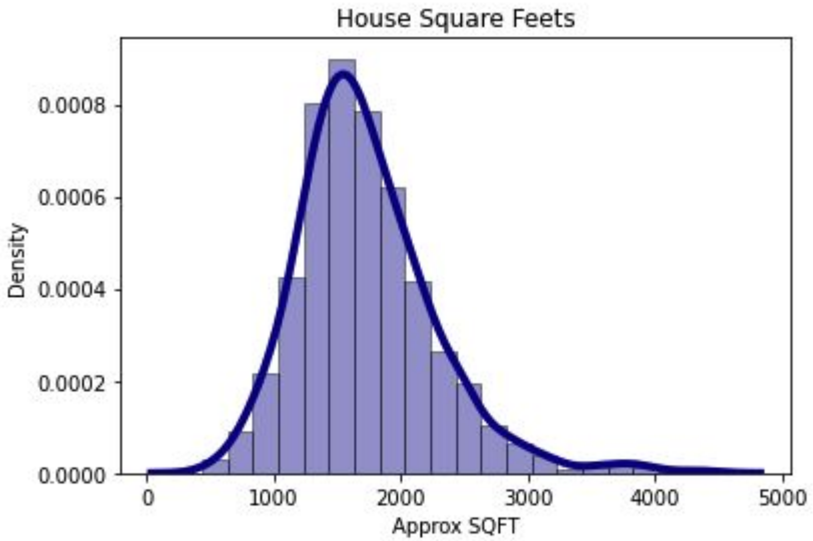
Min:	169000.000
Q1:	390000.000
Median:	462320.000
Q3:	600000.000
Max:	1000000.000
Mean:	515795.279
Mode:	450000.000



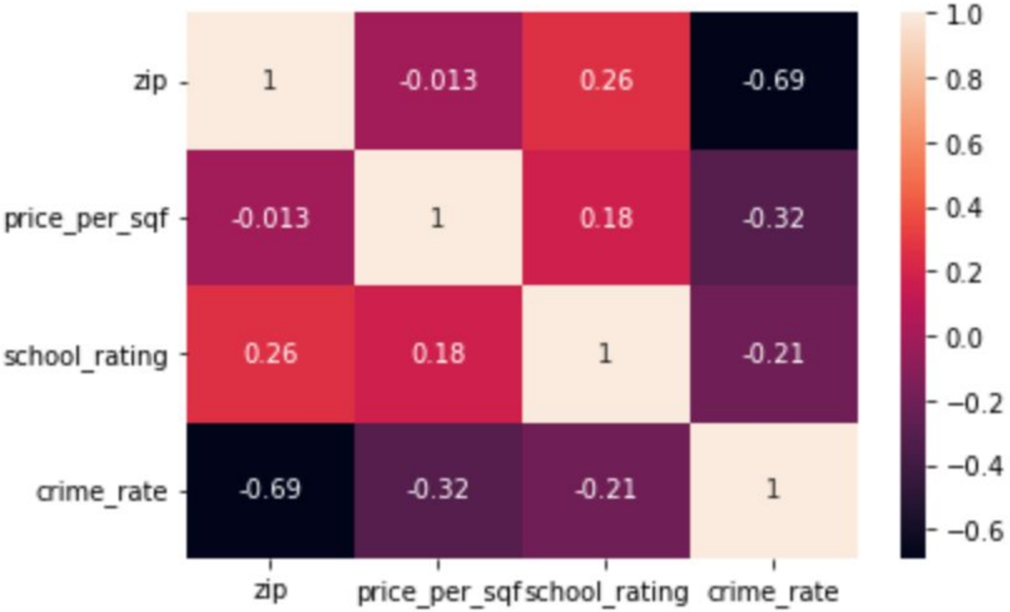
Data Exploration: Descriptive Statistics

A density plot for square footage:

Min:	442.000
Q1:	1372.000
Median:	1651.000
Q3:	2002.000
Max:	4423.000
Mean:	1726.404
Mode:	1260.000



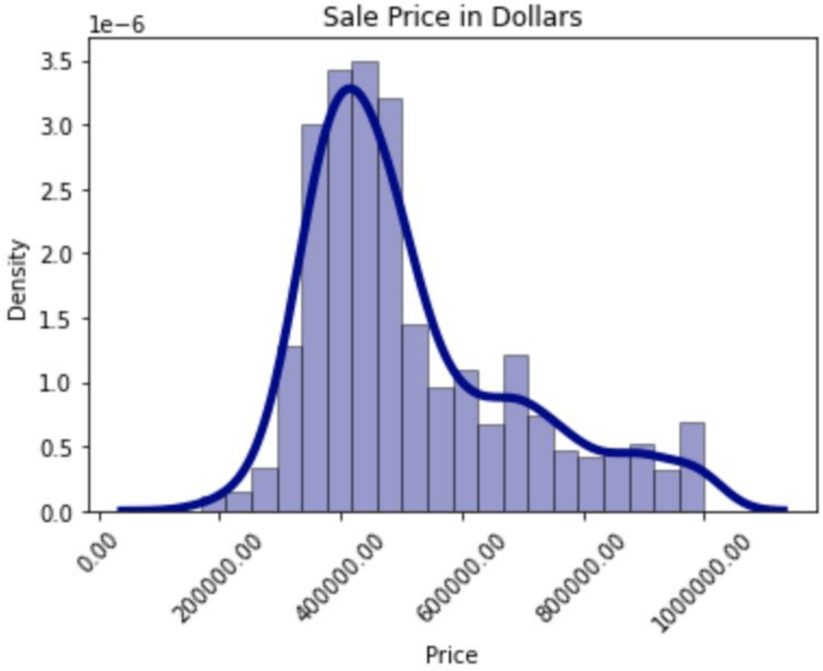
Data Exploration: Descriptive Statistics



Data Exploration: Descriptive Statistics

A density plot for bedrooms:

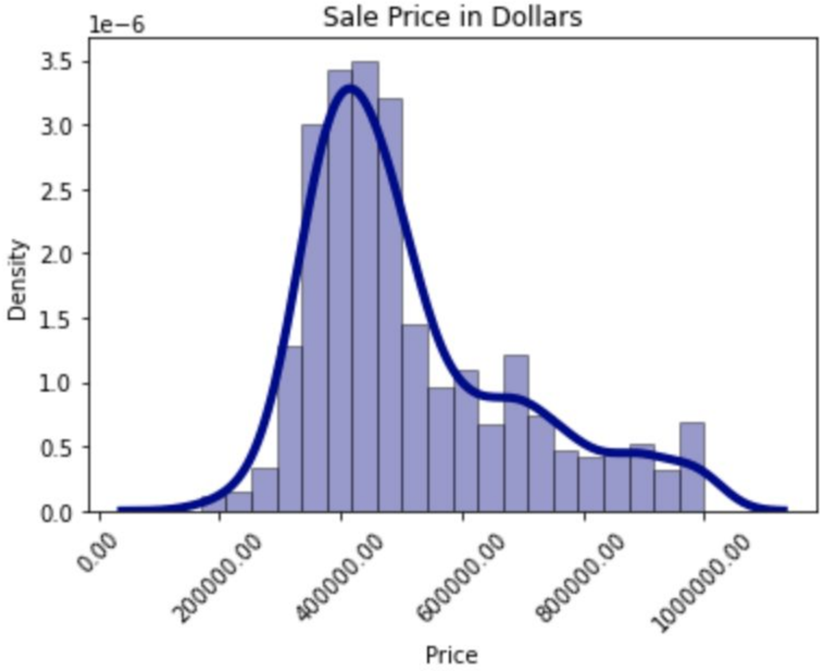
Min:	0.000
Q1:	3.000
Median:	3.000
Q3:	4.000
Max:	12.000
Mean:	3.381
Mode:	3.000



Data Exploration: Descriptive Statistics

A density plot for bathrooms:

Min:	0.000
Q1:	2.000
Median:	2.000
Q3:	2.000
Max:	6.000
Mean:	2.110
Mode:	2.000



Data Exploration: Exploring Models

Multiple linear regression using R:

```r
## statistical tests
ggplot(prop_filter, aes(x=Price)) + geom_density() # visualize distribution using density plot
shapiro.test(prop_filter$Price) # pvalue much less than 0.05, so NOT normal distribution
# note: strong right skew

# anova - considering all features are actually categorical even though they're numbers (sqft is exception)
summary(aov(Price ~ zip, data=prop_filter))
summary(aov(Price ~ zip + sqft, data = prop_filter))
summary(aov(Price ~ zip + sqft + year_built + Bedrooms + Bathrooms, data = prop_filter))

# multiple linear regression
# using only numerical values - that is, not year_built or zip
summary(lm(Price ~ sqft + Bedrooms + Bathrooms, data = prop_filter))
summary(lm(Price ~ sqft + Bedrooms + Bathrooms + age, data = prop_filter))
summary(lm(Price ~ sqft + Bedrooms + Bathrooms + age + ppsf, data = prop_filter)) # is ppsf redundant?

summary(lm(Price ~ sqft + Bedrooms + Bathrooms + age + zip, data = prop_filter))
```

Data Exploration: Exploring Models

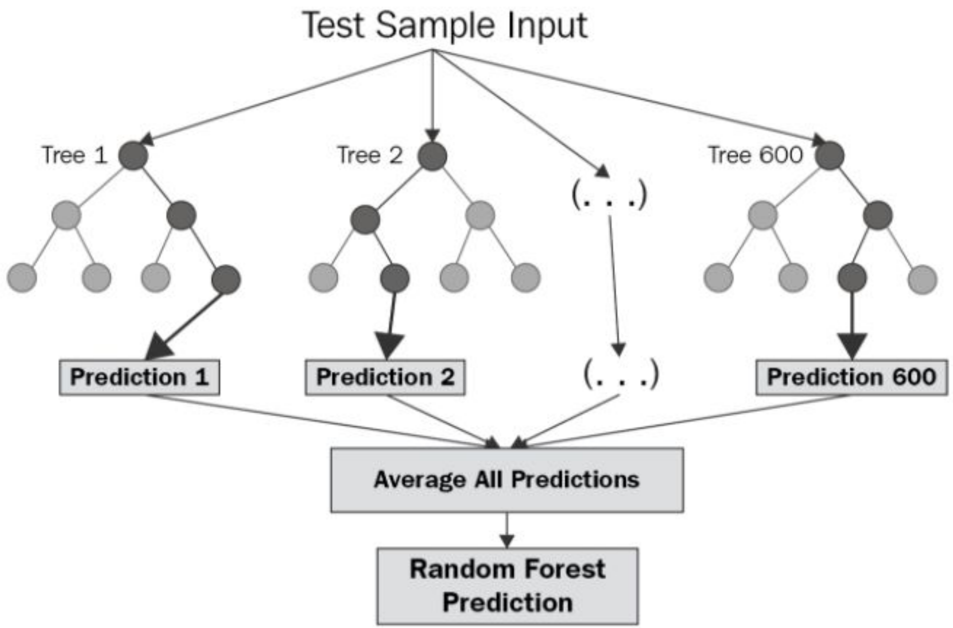
Neural Network using Python's TensorFlow in a Jupyter Notebook

In [36]:

```
# Create a method that creates a new Sequential model with hyperparameter options
def create_model(hp):
    nn_model = tf.keras.models.Sequential()

    # Allow kerastuner to decide which activation function to use in hidden layers
    activation = hp.Choice('activation',['relu','tanh'])

    # Allow kerastuner to decide number of neurons in first layer
    nn_model.add(tf.keras.layers.Dense(units=hp.Int('first_units',
        min_value=1,
        max_value=30,
        step=5), activation=activation, input_dim=len(X_train[0])))

    # Allow kerastuner to decide number of hidden layers and neurons in hidden layers
    for i in range(hp.Int('num_layers', 1, 5)):
        nn_model.add(tf.keras.layers.Dense(units=hp.Int('units_' + str(i),
            min_value=1,
            max_value=30,
            step=5),
            activation=activation))

    nn_model.add(tf.keras.layers.Dense(units=1, activation="relu"))

    # Compile the model
    nn_model.compile(loss="binary_crossentropy", optimizer='adam', metrics=["accuracy"])

    return nn_model
```

Data Exploration: Exploring Models

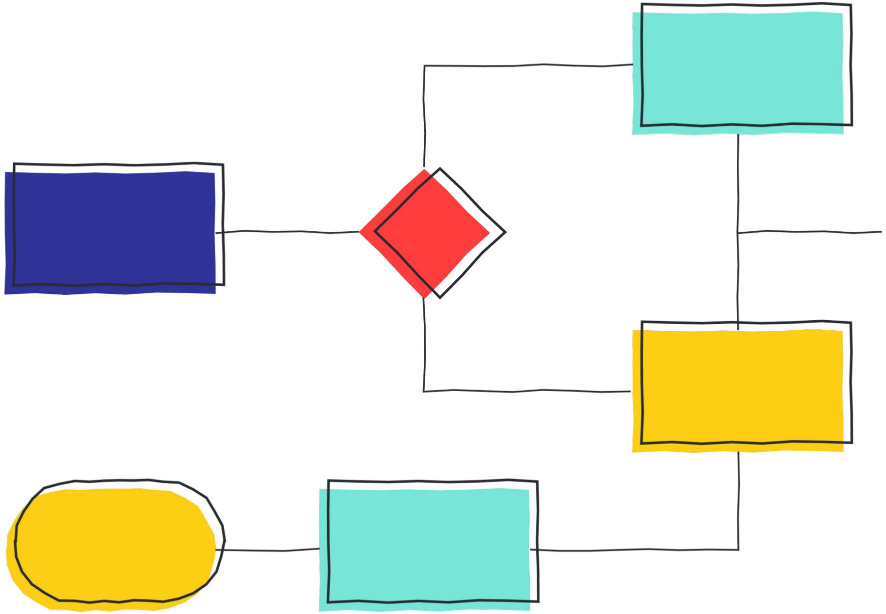
Given the dataset, which contains both numerical and categorical data, and the purpose of the project, which seeks to predict a house's price given various features, we landed on a Random Forest Regressor Model as most appropriate for this project.



Tools Used

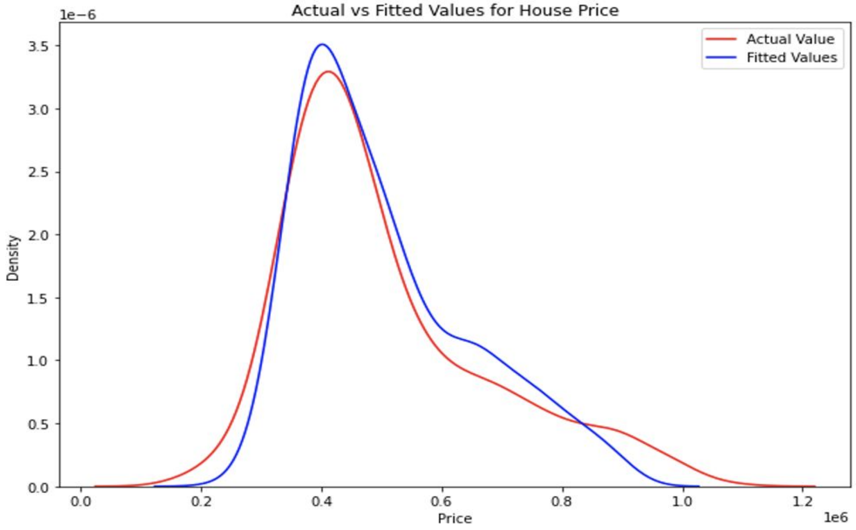
Workflow

1. Import Dependencies
2. Import Data
3. Data Cleaning
4. Finding Best Parameters
5. Database - SQLite
6. Random Forest Model Creation
7. Export necessary information for storyboard / dashboard

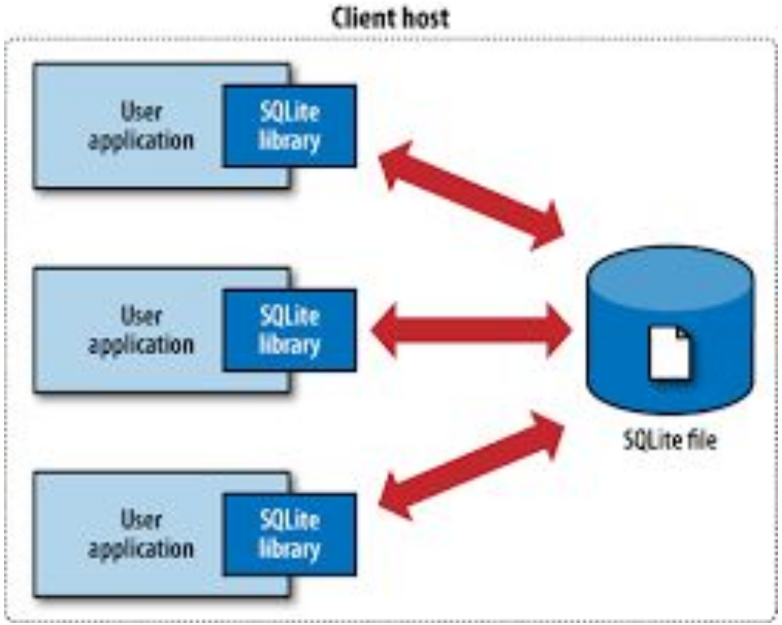


Random Forest Model

Random Forest Model

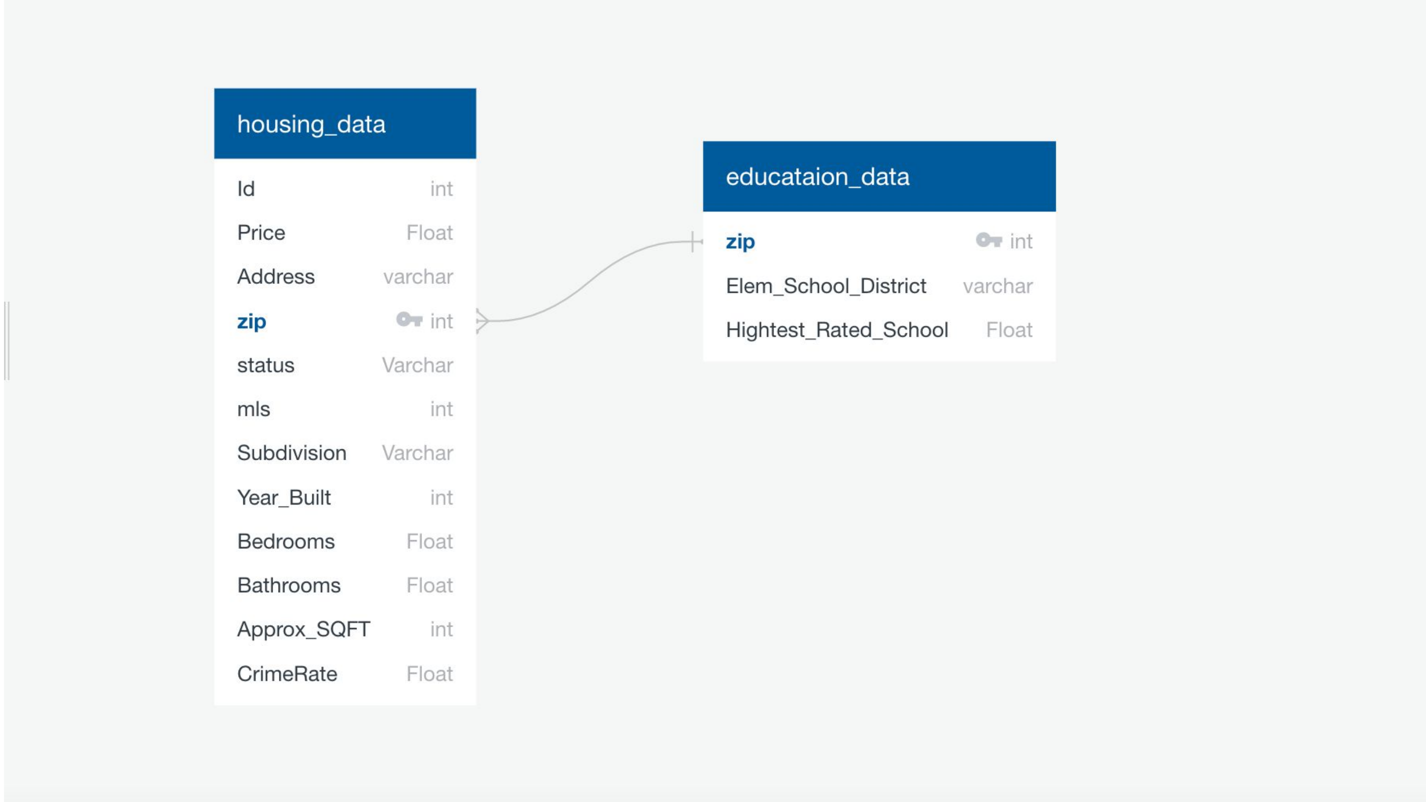


SQLite



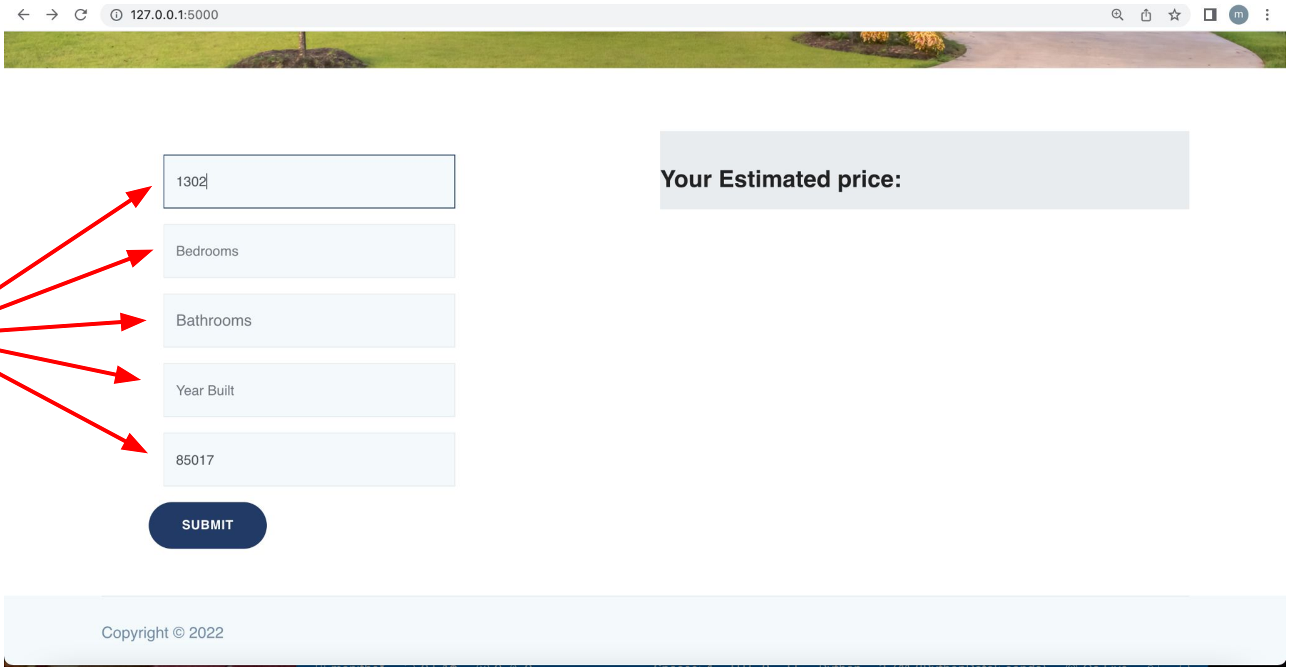
For this project, SQLite will serve to hold our final combined dataset in order to provide the data for visualizations on the dashboard in conjunction with the primary purpose - a predicted home price.

Entity Relationship Diagram

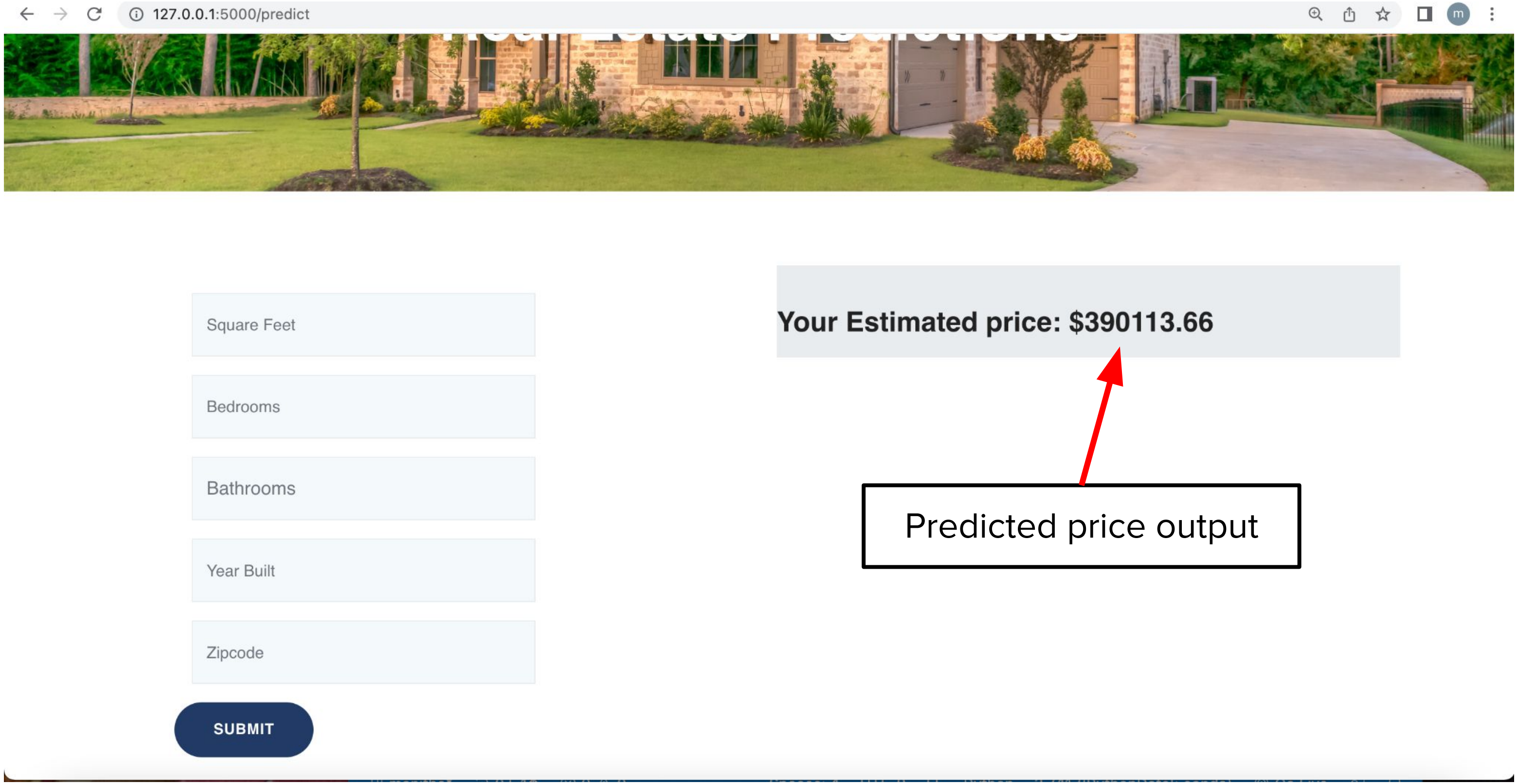


Dashboard

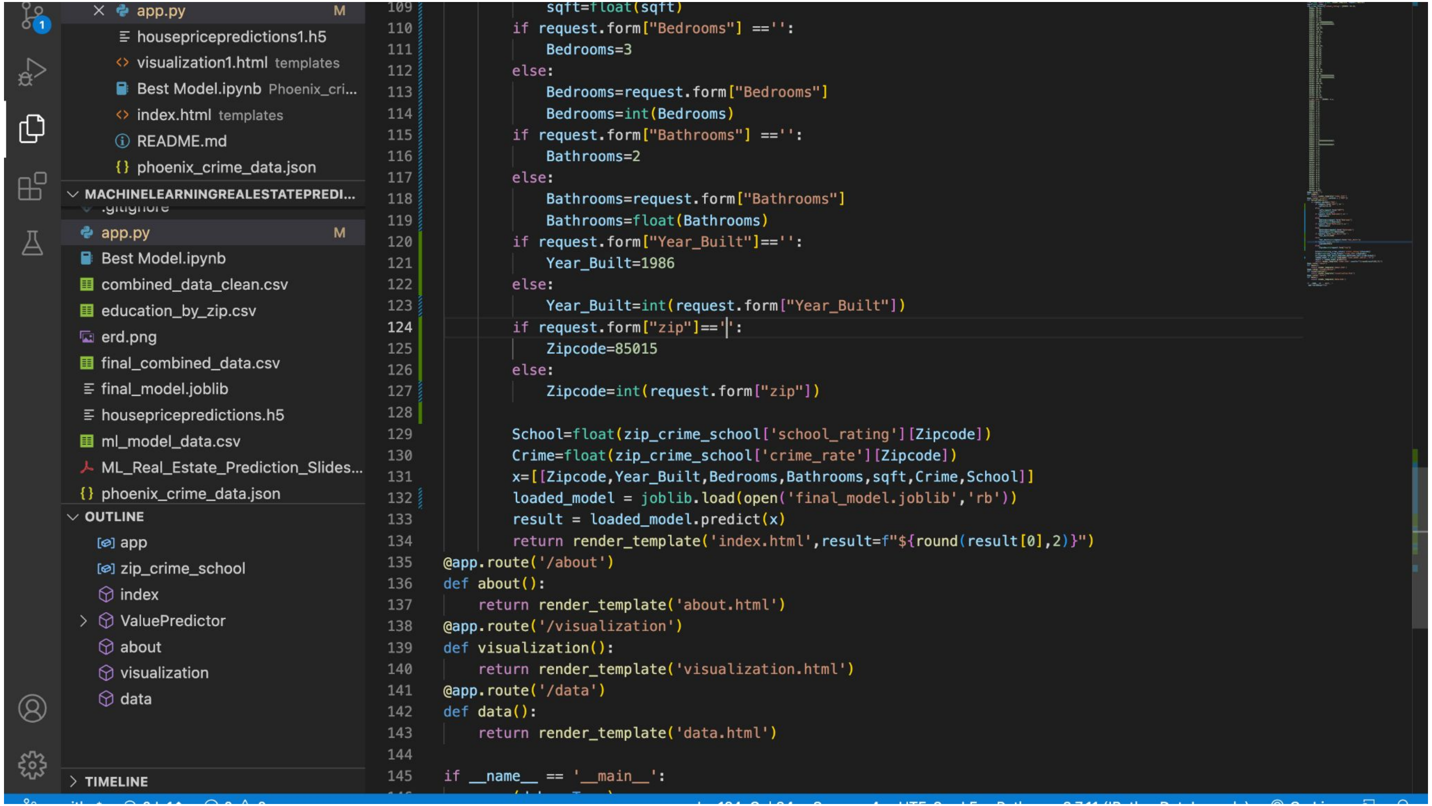
Input boxes



Dashboard


127.0.0.1:5000/predict
Square Feet
Bedrooms
Bathrooms
Year Built
Zipcode
SUBMIT
Your Estimated price: $390113.66
Predicted price output

Dashboard



app.py M
housepricepredictions1.h5
visualization1.html templates
Best Model.ipynb Phoenix_cri...
index.html templates
README.md
phoenix_crime_data.json
MACHINELEARNINGREALESTATEPREDI...
app.py M
Best Model.ipynb
combined_data_clean.csv
education_by_zip.csv
erd.png
final_combined_data.csv
final_model.joblib
housepricepredictions.h5
ml_model_data.csv
ML_Real_Estate_Prediction_Slides...
phoenix_crime_data.json
OUTLINE
app
zip_crime_school
index
ValuePredictor
about
visualization
data
TIMELINE
sqft=float(sqft)
if request.form["Bedrooms"] =='':
Bedrooms=3
else:
Bedrooms=request.form["Bedrooms"]
Bedrooms=int(Bedrooms)
if request.form["Bathrooms"] =='':
Bathrooms=2
else:
Bathrooms=request.form["Bathrooms"]
Bathrooms=float(Bathrooms)
if request.form["Year_Built"]=='':
Year_Built=1986
else:
Year_Built=int(request.form["Year_Built"])
if request.form["zip"]=='':
Zipcode=85015
else:
Zipcode=int(request.form["zip"])
School=float(zip_crime_school['school_rating'][Zipcode])
Crime=float(zip_crime_school['crime_rate'][Zipcode])
x=[[Zipcode,Year_Built,Bedrooms,Bathrooms,sqft,Crime,School]]
loaded_model = joblib.load(open('final_model.joblib','rb'))
result = loaded_model.predict(x)
return render_template('index.html',result=f"${round(result[0],2)}")
@app.route('/about')
def about():
return render_template('about.html')
@app.route('/visualization')
def visualization():
return render_template('visualization.html')
@app.route('/data')
def data():
return render_template('data.html')
if __name__ == '__main__':