**Machine Learning Model Mockup**

**Team Test Data**

**Design Study**

Identify question to be answered: which factors influence home values

Identify target variable: home values

Identify model: RandomForestRegressor, HistGradientBoostingRegressor (secondary)

**Concept Development**

This study is a review of housing values to discover trends and influences related to house features (age of the house, number of rooms, number of bedrooms, location), and external community characteristics (weather, employment rates).

**Data Selection Process**

The team considered multiple factors to include in the study. In addition to the features above, it researched availability of community crime statistics and economic indicators. The deciding factor of whether to include more variables was based on the ease with which data could be merged into the larger dataset. To expedite the model development, the team decided to streamline the study process and focus on fewer variables. However, depending on model outcomes, the team has opted to possibly include other features in later phases. The initial geographical range was decided to be California.

**The Database**

***Description***

After considering several data sources for the housing data, the team identified the California Housing Prices database from Kaggle, (details below) as the main dataset. This dataset is comprehensive, wide-ranging, saturated in geographic area, and includes geographical location coordinates which can link to a wide range of other data sources. The external data for county employment figures were derived from census data (Census.gov) and weather from openweathermap.org, both called using APIs. The population information is the Kaggle California cities dataset. After cleaning, restructuring, refining and merging the individual datasets, these four datasets became the production database and subsequently housed in AWS.

***Component datasets: details***

* Census.csv:

1990 Census data on communities

Selected features (3):

* + counties
  + Employees
  + Establishments

Observations: 60

* Housing.csv:

1999 Census data on housing in communities in California

Data is gathered by block

Features (11):

|  |  |
| --- | --- |
| longitude | A measure of how far west a house is; a higher value is farther west |
| latitude | A measure of how far north a house is; a higher value is farther north |
| housingMedianAge | Median age of a house within a block; a lower number is a newer building |
| totalRooms | Total number of rooms within a block |
| totalBedrooms | Total number of bedrooms within a block |
| population | Total number of people residing within a block |
| households | Total number of households, a group of people residing within a home unit, for a block |
| medianIncome | Median income for households within a block of houses (measured in tens of thousands of US Dollars) |
| medianHouseValue | Median house value for households within a block (measured in US Dollars) |
| oceanProximity | Location of the house w.r.t ocean/sea |

Observations: 20,641

* Weather data:

Weather for specific date called through weather API

Features (5):

* + Max Temp
  + Humidity
  + Cloudiness
  + Wind Speed
  + Description

Observations: 20,433 (after merge with cleaned housing dataset)

* Population data:

Population information by county and city

Features (7):

* + County
  + City
  + Incorporation\_date
  + pop\_april\_1980
  + pop\_april\_1990
  + pop\_april\_2000
  + pop\_april\_2010

Observations: 455

***Structuring and Cleaning***

Creating common columns to link the datasets was the first step. The housing file did not include any city names, only the geo coordinates. The other datasets were identified by city and county. The initial transformation added the specific city and county names to the housing dataset by using city.py and the location coordinates to list and append each city name to the housing set.

Graphical user interface

Description automatically generated

Original dataset

Graphical user interface, application

Description automatically generated

Dataset with City Names

***Create Table Structure pgAdmin***

The main database was structured according to this ERD:

Diagram

Description automatically generated with medium confidence

Weather, population, and census were joined into the main dataset, clean\_merged\_data.csv.

Output database: clean\_merged\_data.csv

Observations: 11,454

***Data Review***

Review data for normality and cross-influences

Identify outliers

Decide on protocol for missing or outlying values

Identify relevant/desirable features

**Create Model**

***Dependencies***

[NumPy](https://www.askpython.com/python-modules/numpy/python-numpy-module)

[pandas](https://www.askpython.com/python-modules/pandas/python-pandas-module-tutorial)

[matplotlib](https://www.askpython.com/python-modules/matplotlib/python-matplotlib)

Sklearn

***Load and preprocess***

Read dataset into pandas DataFrame

Input data: clean\_merged\_data.csv

1. Pandas read csv clean\_merged\_data.csv
2. Preprocess:
   1. Drop any null values
   2. Drop unnecessary and/or low value columns

Rename columns as necessary

***Split*** data into training and testing (80,20)

1. Set X to independent variables
2. Set y to target variable identified in study design

***Tune model***

Random\_forest\_tuning = RandomForestRegressor(random\_state = )

param\_grid = {

'n\_estimators': [100, 200, 500],

'max\_features': ['auto', 'sqrt', 'log2'],

'max\_depth' : [4,5,6,7,8],

'criterion' :['squared\_error', 'absolute\_error']

}

GSCV = GridSearchCV(estimator=random\_forest\_tuning, param\_grid=param\_grid, cv=5)

GSCV.fit(X\_train, y\_train)

GSCV.best\_params\_

***Fit Random Forest regression to dataset***

First Pass:

* Number of trees: n\_estimators ({n\_estimators})
* Criterion
* Random\_state

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators = 100, random\_state = 0)

regressor.fit(X,y)

***Test prediction***

y\_pred**=**regressor.predict()

y\_pred

***Create visualization graph***

***Interpret graph and test prediction***

***Compute, print, graph feature importances***

features = sorted(zip(X\_train.columns, regressor.feature\_importances\_), key = lambda x: x[1], reverse=True)

features[:11]

features = X.columns.values  
importances = reg.feature\_importances\_  
indices = np.argsort(importances)  
  
plt.title('Feature Importances')  
plt.barh(range(len(indices)), importances[indices], color='#8f63f4', align='center')  
plt.yticks(range(len(indices)), features[indices])  
plt.xlabel('Relative Importance')  
plt.show()

***Adjust model***

N\_estimators

Features

Secondary Model