Enhance IT

Classification and Regression (Raw-Houses)

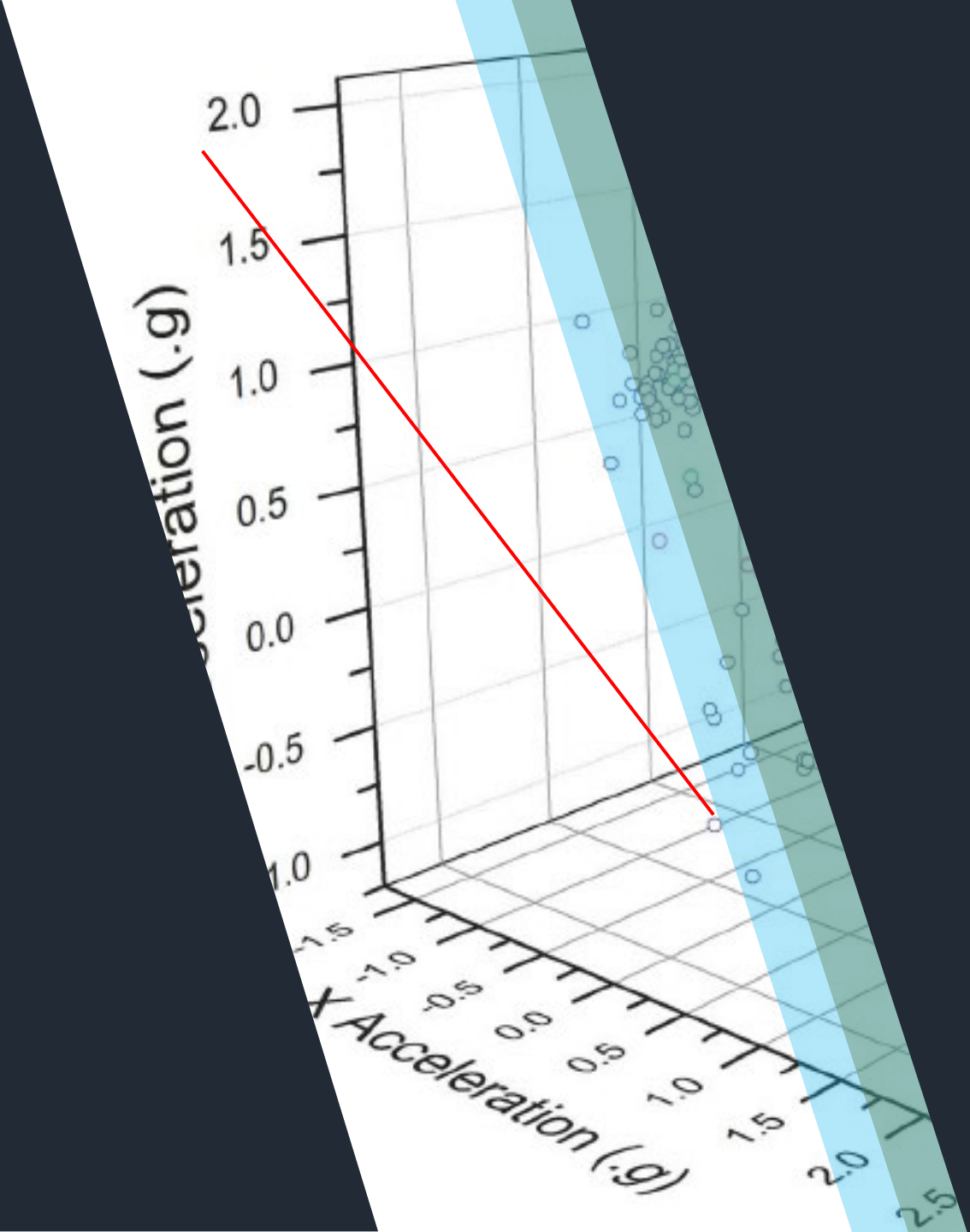
AUTHOR

David Antonio González Gómez

David.Gonzalez@enhanceit.us

TRAINING COURSE / 3rd Delivery

Atlanta GA, December 2021



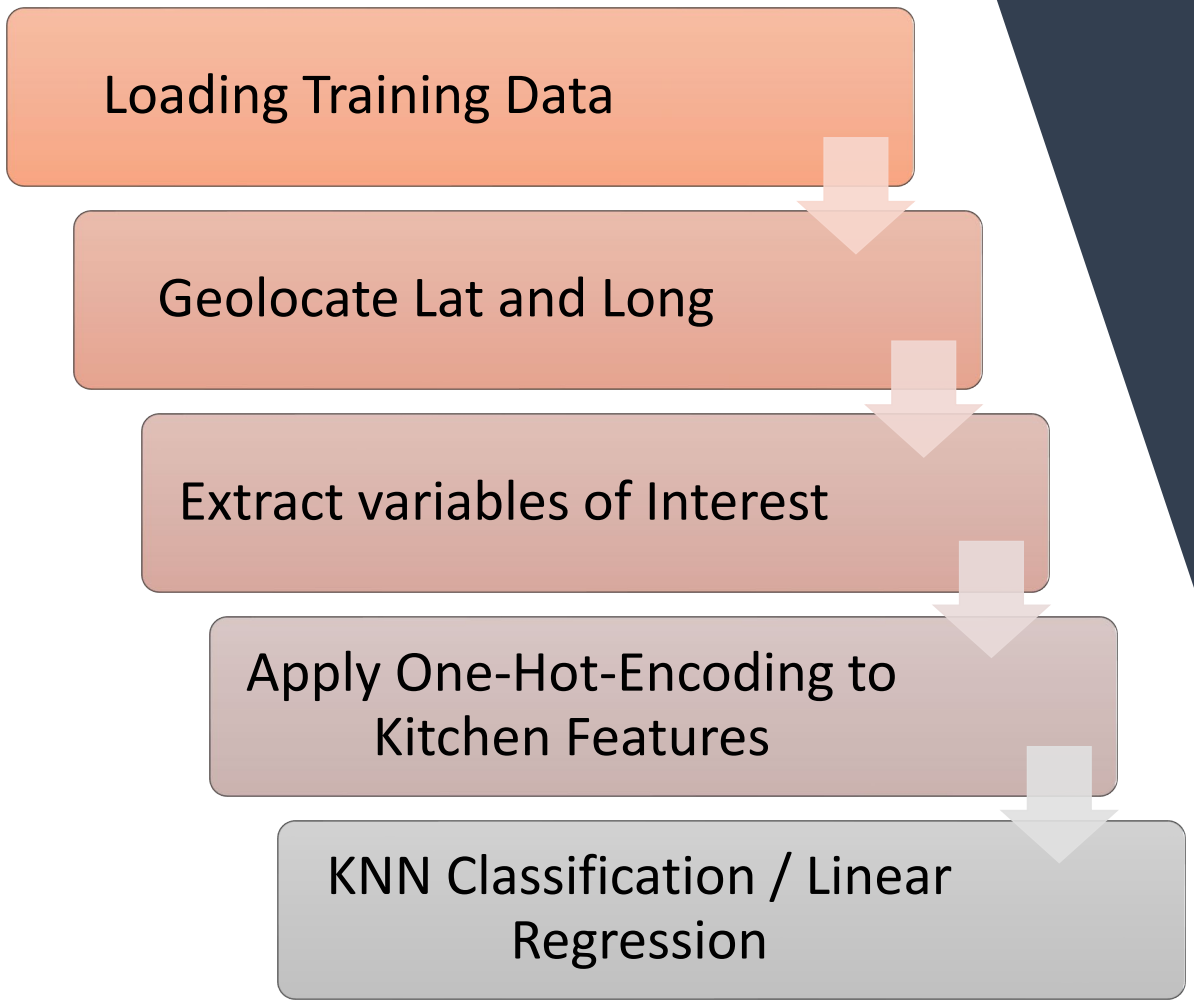
Objective

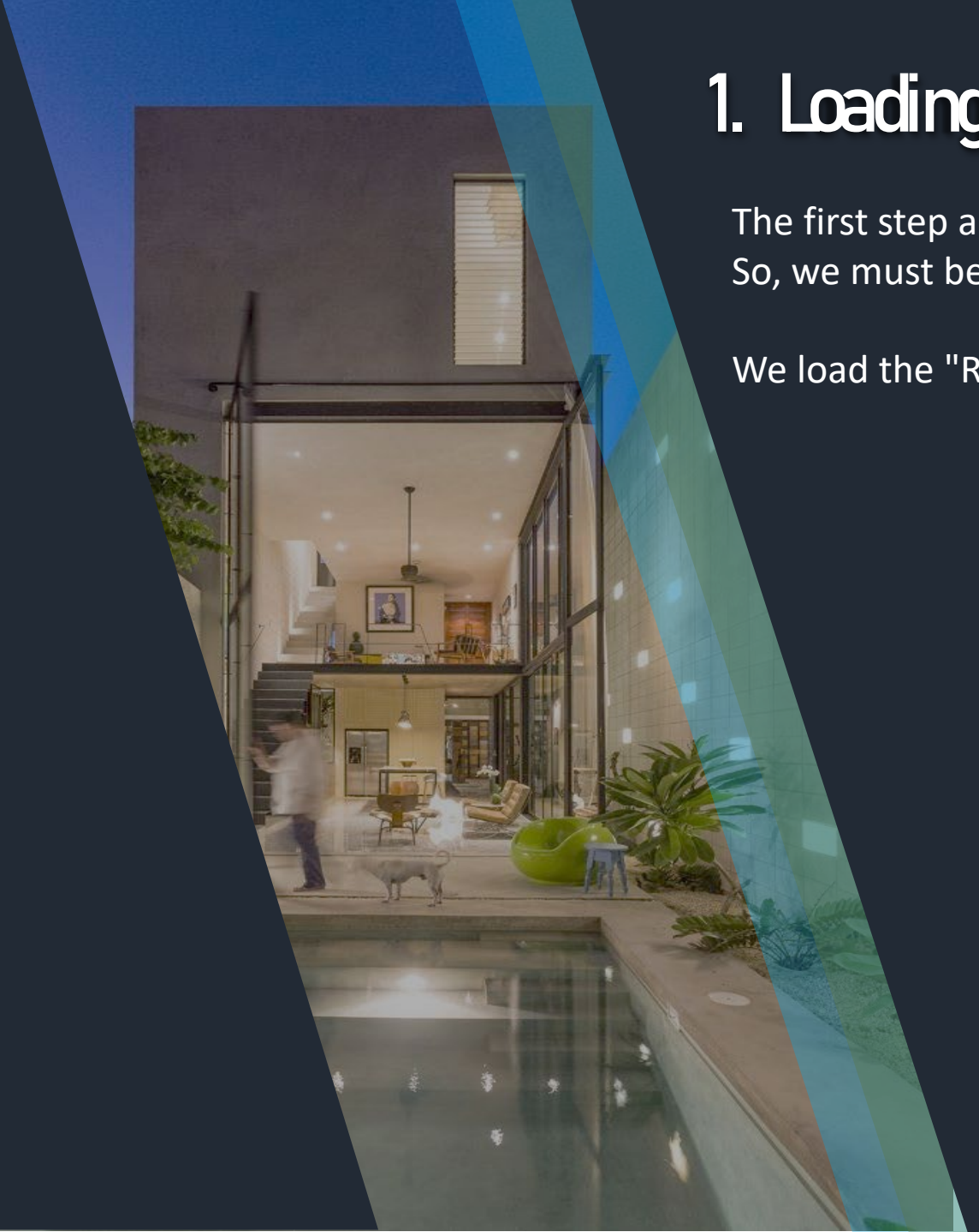
Create a use case using the RAW HOUSES dataset and implement a KNN classifier with LINEAR REGRESSION to recognize a specific task.

Development Steps

In this project a database for Raw-Houses is presented. The objective is to do a deep exploration to analyze the houses that do not have their kitchens equipped using KNN (k nearest neighbors) and Linear Regression. Initially, we have a database with **5,000 observations and 16 features**.

The document is divided into the following sections for data processing, as shown below:





1. Loading Database

The first step and the most basic step is to understand the variables in the data set. So, we must be quite sure about what type of data is being worked on.

We load the "Raw Houses" data as a Pandas data frame and look at the columns:

```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Enhance IT/H3: Data
```

```
print (data.head())
print (data.shape)
print(data.columns)
```

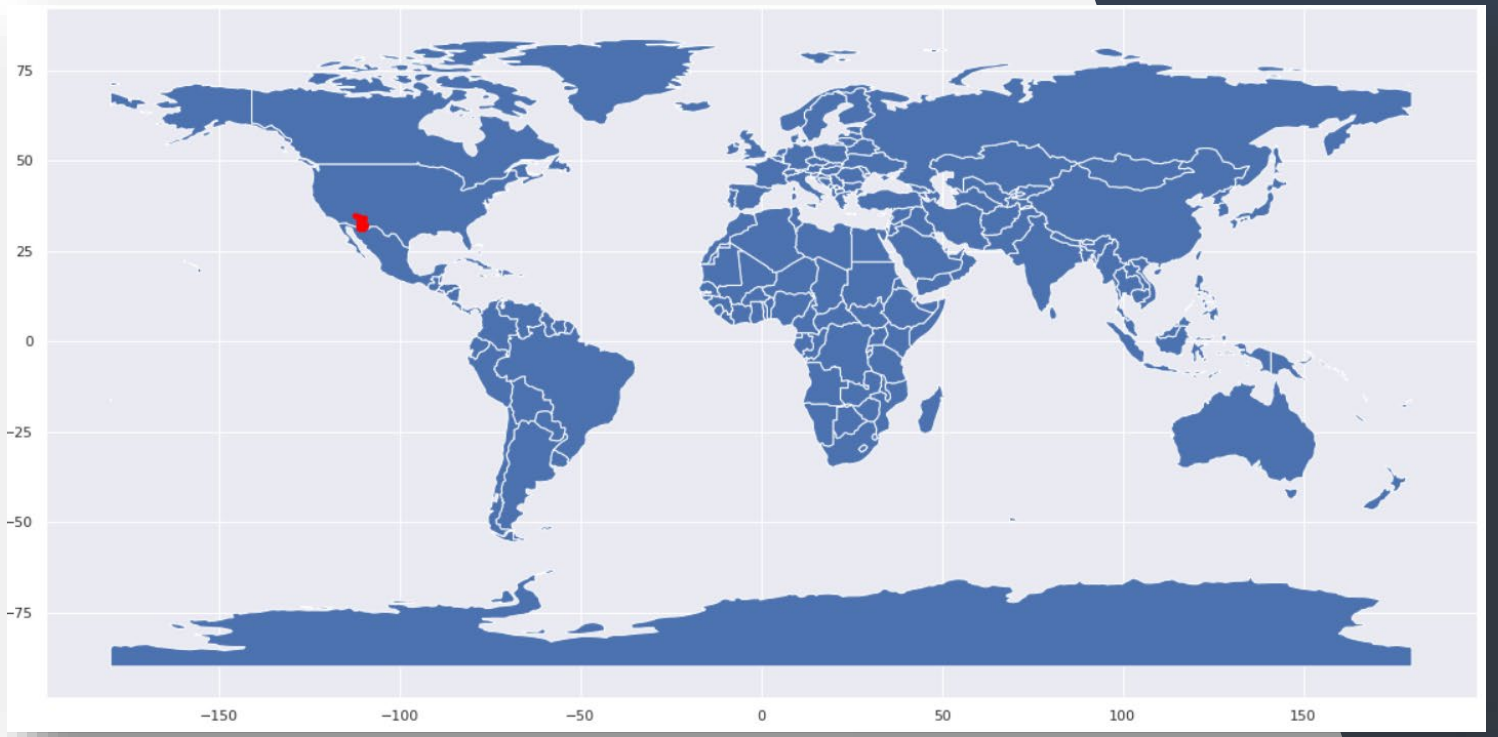
```
        MLS  sold_price  ...                floor_covering   HOA
0  21530491   5300000.0  ...            Mexican Tile, Wood     0
1  21529082   4200000.0  ...          Natural Stone, Other     0
2   3054672   4200000.0  ...     Natural Stone, Other: Rock  None
3  21919321   4500000.0  ...  Ceramic Tile, Laminate, Wood  None
4  21306357   3411450.0  ...              Carpet, Concrete    55

[5 rows x 16 columns]
(5000, 16)
Index(['MLS', 'sold_price', 'zipcode', 'longitude', 'latitude', 'lot_acres',
       'taxes', 'year_built', 'bedrooms', 'bathrooms', 'sqrt_ft', 'garage',
       'kitchen_features', 'fireplaces', 'floor_covering', 'HOA'],
      dtype='object')
```

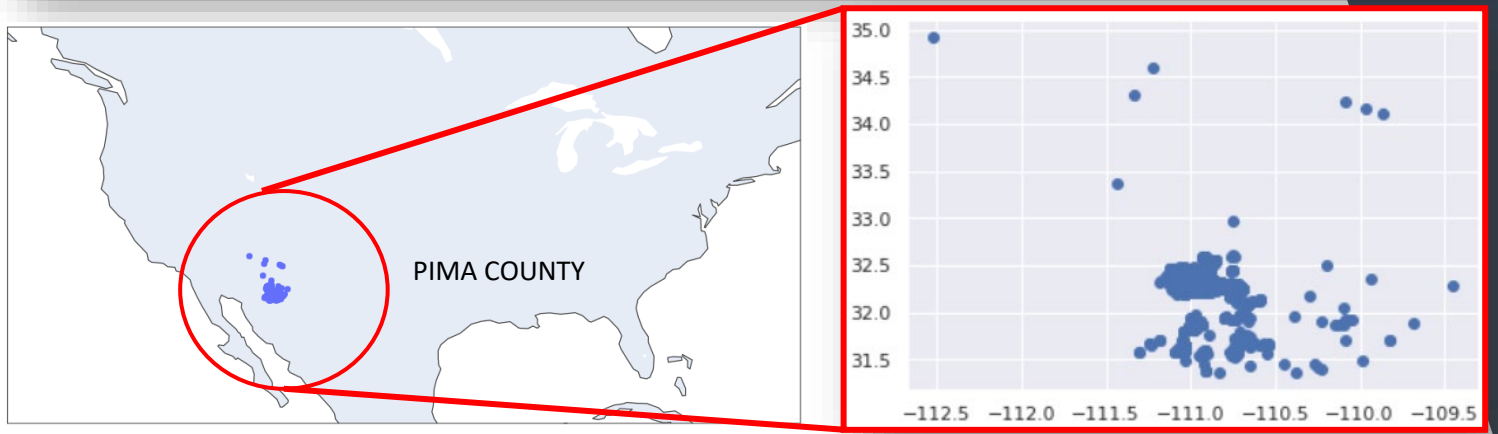
Longitude and Latitude display

It is important to carry out this type of analysis to know the area / region / country / or state in which we are working.

Use Geoplot as it is a high-level Python geospatial plotting library. It is an extension of cartopy and matplotlib that makes mapping easy: as seaborn for geospatial.

We can see that our 5,000 features are located within Arizona.





Variables Assign

The next step is to clean the data of the redundancies that can be irregularities in the data that can be some variables or some columns that are not necessary to make our conclusions or interpretations. We can just remove them or they are outliers that can cause noise in the data.

- **X_Train:** Contains a Matrix / lists with 5 features (Sold Price, Zip Code, Long, Lat and Lot Acres).

- **Y_Train**: Contains exclusively a column that corresponds the labels (Kitchen Features)

```
print(CleanData)
print(CleanData.columns)
print(CleanData.dtypes)
```

```
      sold_price  ...                                   kitchen_features
0      5300000.0  ...            Dishwasher, Freezer, Refrigerator, Oven
1      4200000.0  ...                        Dishwasher, Garbage Disposal
2      4200000.0  ...          Dishwasher, Garbage Disposal, Refrigerator
3      4500000.0  ...  Dishwasher, Double Sink, Pantry: Butler, Refri...
4      3411450.0  ...  Dishwasher, Garbage Disposal, Refrigerator, Mi...
...          ...  ...                                                ...
4995    495000.0  ...  Dishwasher, Double Sink, Garbage Disposal, Gas...
4996    550000.0  ...  Dishwasher, Double Sink, Electric Range, Garba...
4997    475000.0  ...  Dishwasher, Electric Range, Island, Refrigerat...
4998    550000.0  ...  Dishwasher, Double Sink, Garbage Disposal, Gas...
4999    450000.0  ...  Compactor, Dishwasher, Double Sink, Island, Ap...

[5000 rows x 6 columns]
Index(['sold_price', 'zipcode', 'longitude', 'latitude', 'lot_acres',
       'kitchen_features'],
      dtype='object')
sold_price          float64
zipcode               int64
longitude           float64
latitude            float64
lot_acres           float64
kitchen_features     object
dtype: object
```

CLINNG THE COLUMN Kitchen_Features

```
Dishwasher, Garbage Disposal, Refrigerator, Microwave, Oven                          1719
Dishwasher, Garbage Disposal, Microwave, Oven                                         270
Compactor, Dishwasher, Garbage Disposal, Refrigerator, Microwave, Oven                189
Dishwasher, Garbage Disposal, Refrigerator, Oven                                      181
Dishwasher, Freezer, Garbage Disposal, Refrigerator, Microwave, Oven                  127
                                                                                     ...
Dishwasher, Garbage Disposal, Gas Range, Pantry: Walk-In, Refrigerator, Countertops: Grar    1
Desk, Dishwasher, Electric Range, Garbage Disposal, Pantry: Cabinet, Refrigerator, Applia    1
Dishwasher, Double Sink, Garbage Disposal, Gas Range, Pantry: Cabinet, Refrigerator, Appl    1
Desk, Dishwasher, Electric Range, Freezer, Gas Range, Indoor Grill, Island, Prep Sink, Re    1
Dishwasher, Double Sink, Freezer, Garbage Disposal, Gas Range, Island, Pantry: Walk-In, F    1
Name: kitchen_features, Length: 1872, dtype: int64
```

Appliance Color: Stainless — REMOVE
Countertops: granite — REMOVE
Missing: wine cooler — REMOVE
Microwave: yes — JUST: MICROWAVE

```
['dishwasher,freezer,refrigerator,oven' 'dishwasher,garbagedisposal'
 'dishwasher,garbagedisposal,refrigerator' ...
 'dishwasher,electricrange,island,refrigerator,reverseosmosis,applia
 'dishwasher,doublesink,garbagedisposal,gasrange,pantrycabinet,appli
 'compactor,dishwasher,doublesink,island,appliancecolorstainless']
```

ONE-HOT-ENCODING: Kitchen_Features

A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

	Refri	DishWa	Microwa	Freez	TOTAL
H1	1	0	1	1	3
H2	1	0	0	1	2
H3	1	1	1	1	4
…	…	…	…	…	…
H5,000	1	0	0	0	1

1,872 ⇨ Max 23, Min 1

KNN Classification

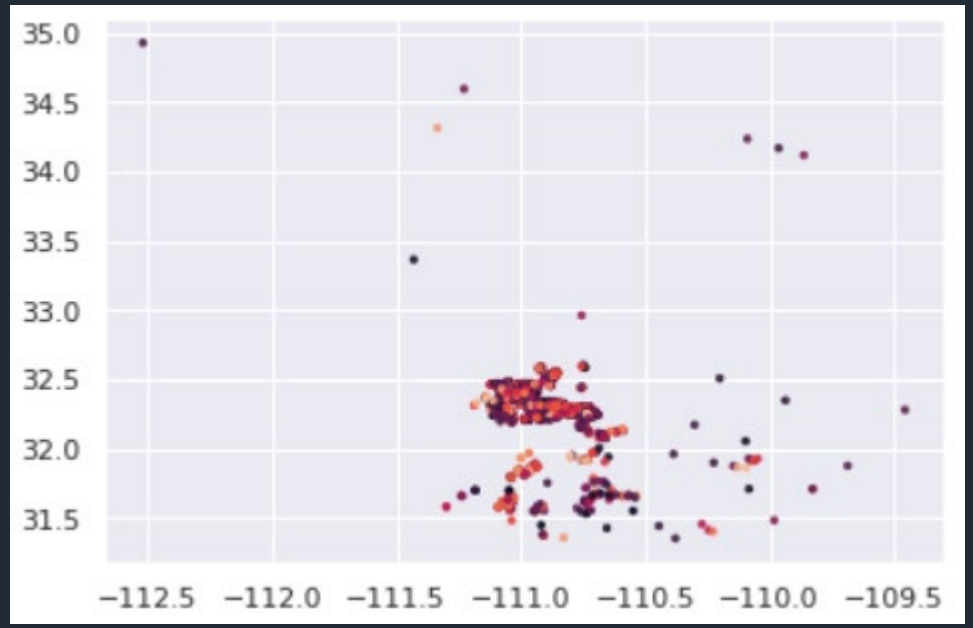
```
class KNNClassifier():

  def fit(self, X, y):
    self.X=X
    self.y=y

  def predict(self, X, K, epsilon=1e-
    N = len(X)
    y_hat = np.zeros(N)

    for i in range(N):
      dist2 = np.sum((self.X-X[i])**2, ax
      idxt = np.argsort(dist2)[:K]
      gamma_k = 1/(np.sqrt(dist2[idxt]+eps
      y_hat[i] = np.bincount(self.y[idxt],

    return y_hat
```

```
[530] def accuracy(y,y_hat):
        return np.mean(y==y_hat)
```

```
[531] accuracy(y_train,y_hat_train)

  0.9548
```

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression.

It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.



FINAL RESULTS

Finally, we join our two databases (training and testing) to calculate the precision of our model:

```
[237] def accuracy(y,y_hat):
        return np.mean(y==y_hat)
```

```
[238] accuracy(y_train,y_hat_train)
```

```
0.9478
```

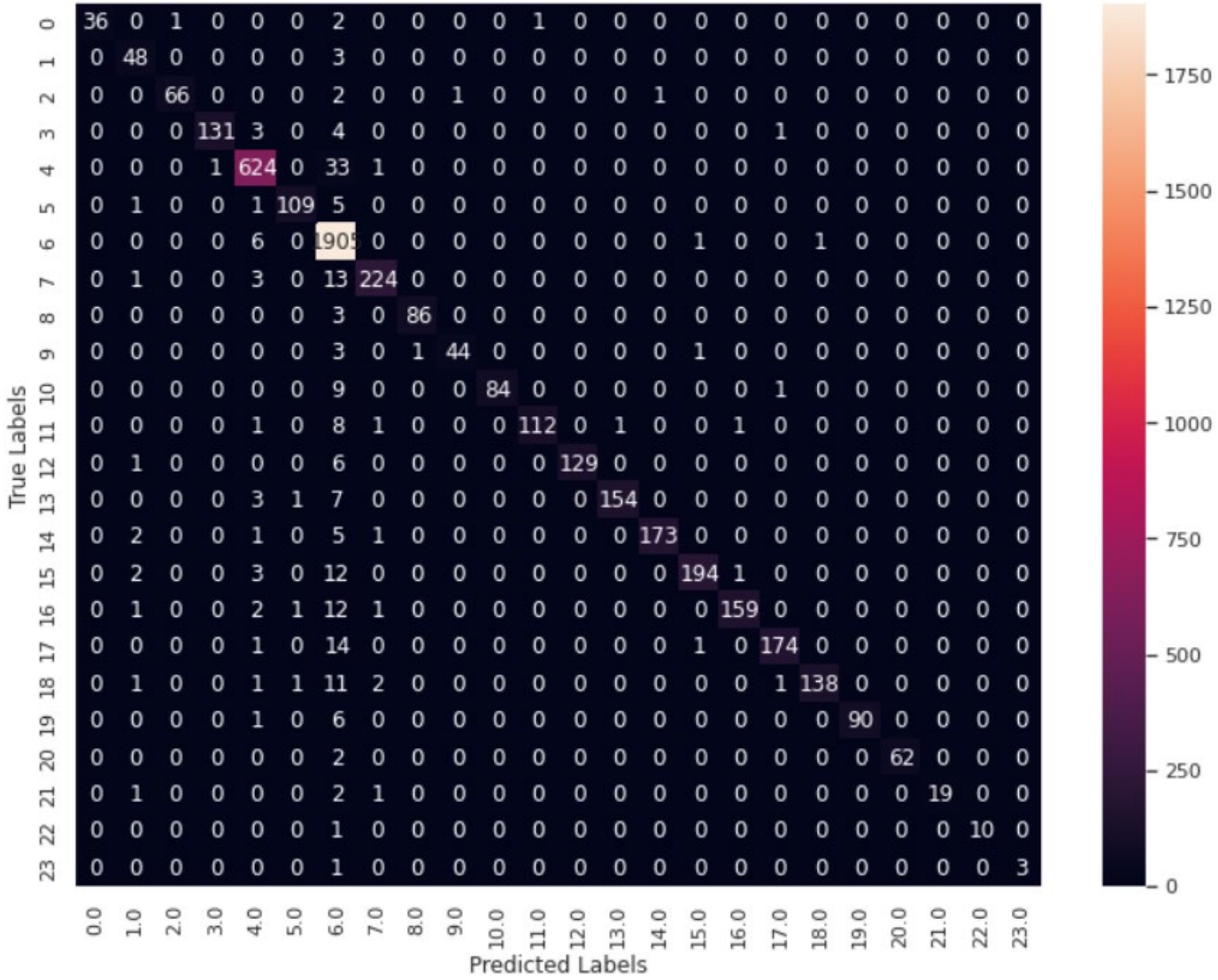
5. Confusion Matrix

When we get the data, after data cleaning, pre-processing, and wrangling, the first step we do is to feed it to an outstanding model and of course, get output in probabilities. But hold on! How in the hell can we measure the effectiveness of our model. Better the effectiveness, better the performance, and that is exactly what we want.

And it is where the Confusion matrix comes into the limelight. Confusion Matrix is a performance measurement for machine learning classification.

```
[[939   0   9   6   0   5   3   1  17   0]
 [  0 546  50   4   8   0   7   0 520   0]
 [  3   0 984  10   1   0   1   3  30   0]
 [  3   0  24 908   1  10   0   4  57   3]
 [  0   0  24   1 923   0   1   5  22   6]
 [  4   0   9  42   2 748   6   1  76   4]
 [ 10   0  13   0   5  21 880   0  29   0]
 [  0   0  17  12  24   1   0 903  32  39]
 [  6   0  20  23   3   6   0   4 910   2]
 [  4   0   6  13  36   1   0  23  52 874]]
```

We can see that 1,905 houses (almost 2,000) just have 6 kitchen items. The question is: Where is the correct place to locate a Store focus in that houses?

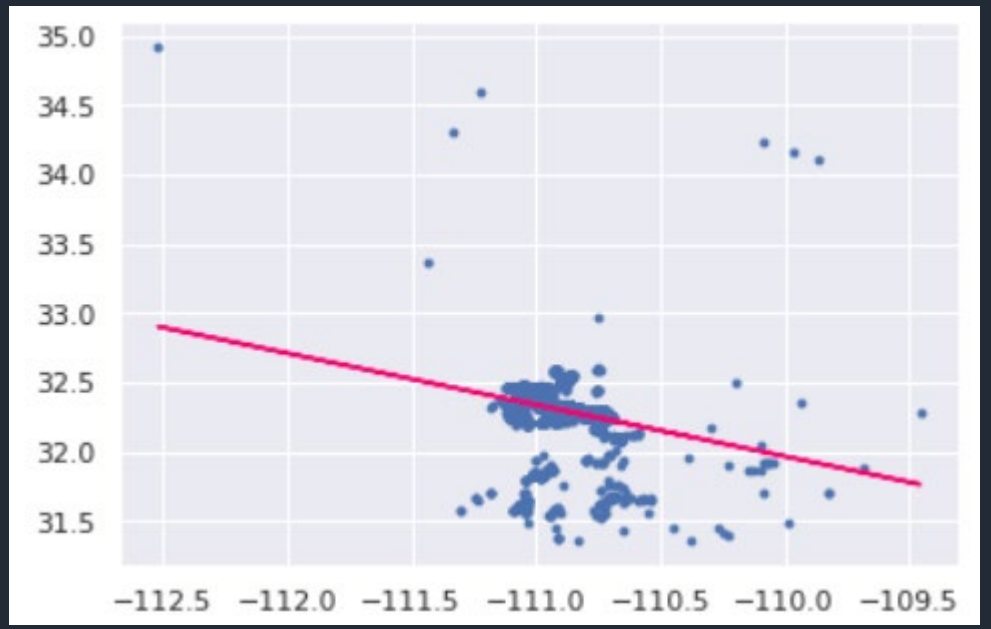


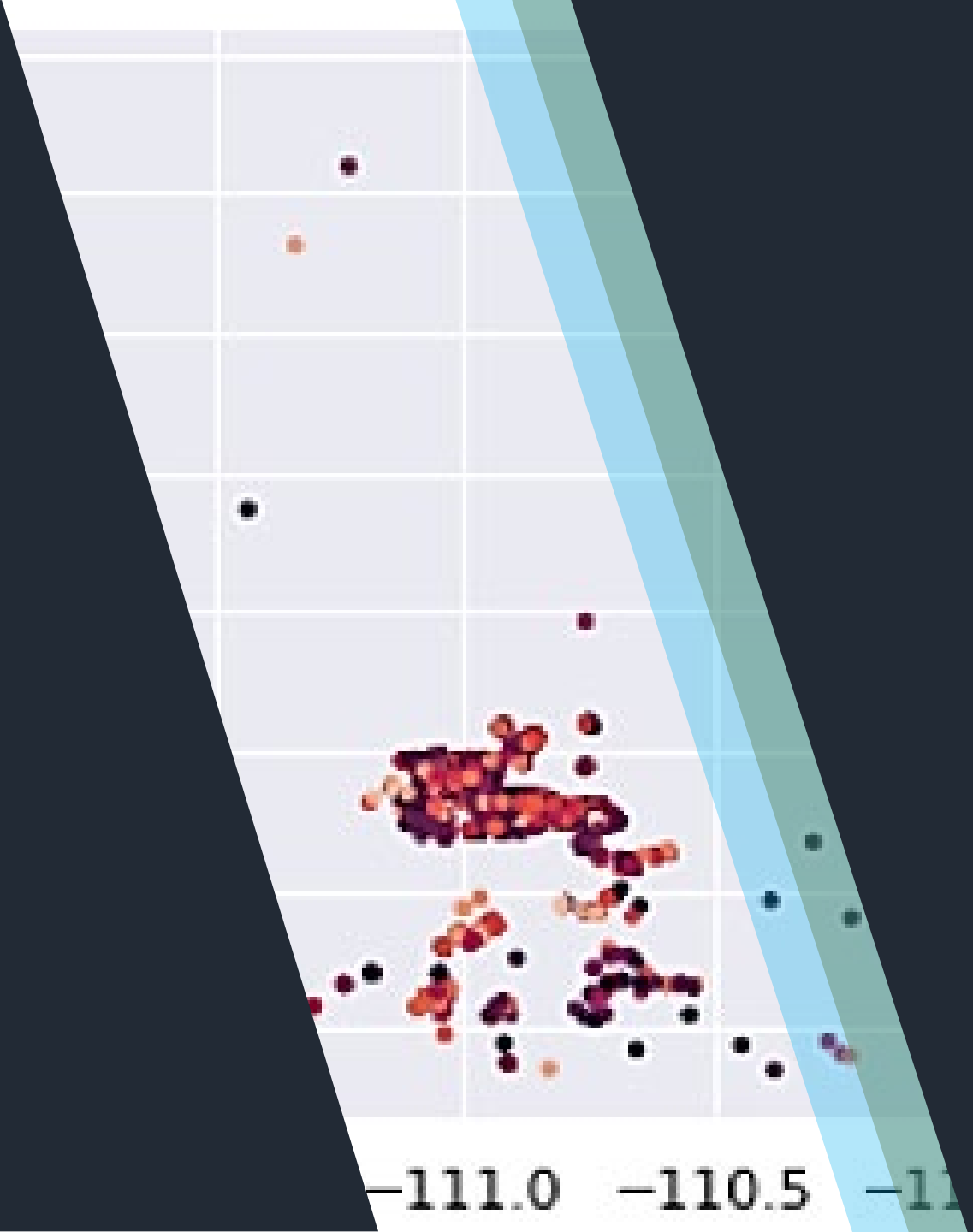
Simple Linear Regression

```
class SimpleLinearReg():

  def fit(self, X, y):
    self.y = y
    self.d = np.mean(X**2)-np.mean(X)**2
    self.w0= (np.mean(y)*np.mean(X**2)-np.
    self.w1= (np.mean(X*y)-(np.mean(X)*np.

  def predict(self, X, show=0):
    y_hat = self.w0+self.w1*X

    if show:
      plt.figure()
      plt.scatter(X, self.y, s=8)
      plt.plot(X, y_hat, color="#FF0070")

    return y_hat
```

Regression models describe the relationship between variables by fitting a line to the observed data. Linear regression models use a straight line, while logistic and nonlinear regression models use a curved line. Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.

Simple linear regression is used to estimate the relationship between two quantitative variables.





Conclusion

KNN can handle sorting problems, and naturally it can handle multiple sorting problems. But every time the classification or regression, the training data and the test data have to be counted. If the amount of data is large, the required computing power is greater.

THANKS

Enhance IT

Atlanta GA, December 2021