Real Estate Predictor
Capstone Project

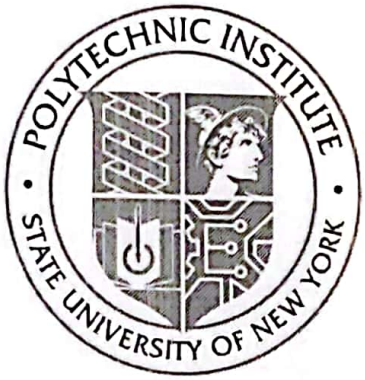
B.S. Capstone

Edward Simpson

Supervised by:

Dr. Bruno Andriamanalimanana

December 2022



Bachelor of Science in Computer and Information Sciences

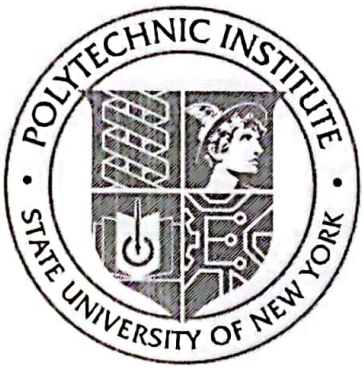
Department of Computer Science

State University of New York Polytechnic Institute

Real Estate Predictor for BS Capstone
a project by Edward Simpson (U00310458),
is approved and recommended for acceptance as a final project in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer and Information Sciences, SUNY Polytechnic Institute.

Dr. Bruno Andriamanalimanana

Date



STUDENT DECLARATION

I declare that this project is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Edward Simpson

Abstract

Due to the vast size and profitability of the real estate market, many want to find ways to efficiently enter it. This requires a lot of research and, even with all that research, it cannot be guaranteed that they are making the most beneficial choice. This program aims to give users an easy-to-use method to analyze real estate data and make the most efficient choices. To achieve this the program does the real estate analysis for them and gives a clear scoring method. The program analyzes multiple areas in the real estate market, specifically around rentals. This also applies predictive methods to generate accurate scores.

Table Of Contents

List of Figures

Chapter 1

Introduction

1.1 Overall

The real estate predictor, REP, is generally a large data processing software. Specifically, one that processes price and rent cost data across major cities in the United States. REP analyzes the recent real estate data, pulled from Zillow [5]. It takes its findings and scores them based on a variety of factors. These factors include: current house cost, projected house cost, rent price, and years to pay. Current house cost is determined based on the average price of a house—a house in the 40^{th} to 60^{th} percentile—across a city. The projected house price is determined by a function built into the program. The function generates the projected cost by analyzing the house price graph for a city over time and applies a time series forecasting model to get the house price in the future. The rent price is then determined as the average rent price, including utilities, for a house in the 40^{th} to 60^{th} percentile range as well. The last function, 'years to pay' is determined by how many years it would take at a projected rent price to fully pay back on the house at the price it was acquired at. The REP takes these four different measures, weights them, and then combines them. With this it gets a score, out of 1000, that measures a cities suitability to buy renters in. Additionally, the REP includes an easy-to-use interface that allows one to individually see and plot all four analyses the REP does. Also, it can plot the overall on an easy to see US Map – to help the viewer try and make some geological inferences on the data.

1.2 Goal

The purpose of the real estate predictor is to provide a naïve user the ability to gain a basic comprehension on the real estate market. This includes average house prices, rent prices, trends, and the geological impact on real estate prices. To do this, it delivers an easy-to-use and

straightforward interface that lets the user see and compare various plots, graphs, and maps. These charts are designed to give the user further awareness on the data.

A secondary goal of the real estate predictor is to show the user the most optimal cities/regions to acquire rental properties. This is determined in the program and given through a numerical scoring system. A map is also given to visually display the data obtained - to allow the user to further see the data, the graph is shown below in figure 1.1. The ability to see and compare regions across the United States lets the user get a better understanding and hopefully propel or improve their real estate craft.

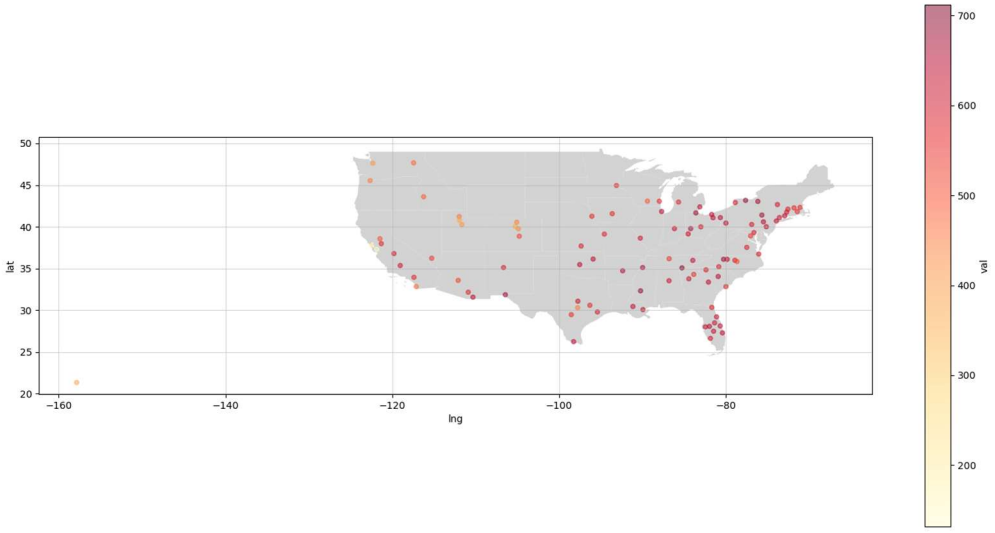


Figure 1.1: Overall Map

Chapter 2

Background

2.1 Real Estate Data

The real estate data included various information related to the individual cities. This information included rent and price data. The data was pulled from the Zillow research website, [5], and is free to download and use. It comes in the form of an excel csv file, grouped by cities. Zillow is a widely used platform for buying, selling, renting, or managing properties based on location. This is achieved mainly through the Zillow website, but they also have agents that can help or do the process for you. Since it is such a widely used platform, it is able to gather lots of data from those that use the website. Of course, this data cannot cover all houses sold or rented – as this data only includes those from Zillow listings

The data on house pricing was derived from listings placed or sold through Zillow. The data is based on individual cities and averaged on a monthly basis from the early 2000s. Also, the listings averaged are limited to only homes in the 40^{th} to 60^{th} percentile. Furthermore, I had to cut the data short because the corresponding rental data was earliest pulled from 2014. Therefore, all plots and graphs use data from 2014 to current.

The rental data, also pulled from Zillow in csv format, included less cities than its corresponding price data and therefore some cities were excluded from the program. Which is why the program is limited to only major cities across the United States. The rental data is also computed on a monthly basis and based on individual cities. It also is based on rental listings posted to Zillow, from homes in the 40^{th} to 60^{th} percentile. Therefore, as with Zillow listings the rental prices include utilities in the price.

Both the rental and price data included irregularities that were fixed manually through excel spreadsheets. The irregularities included missing values, presumably from no data, and different number of cities from both. The sheets were compared to each other to remove cities

that were not listed on both sets of data. Then blank or null values were filled with zero, as to avoid errors from happening when processing the data in the REP.

2.2 Map Data

The data used in order to plot the cities on a map was found from ‘Simple Maps’, [6]. It’s a free and easy to access site that gives an up-to-date file of the latitude, longitude, and population of all census-recognized cities and towns. Simple Maps also offers a few versions that a person can pay for that gives more comprehensive information on cities, but that was not used as it was a tad bit too expensive. It’s format also did not match the format from the data pulled from Zillow. Therefore, a lot of manual changes had to be done to the csv in order to match the data. These changes included using excel to match the csv files and apply a region number to each city that was located in Zillow data. There were only a few cities from the Zillow data whose names were not matched in the map data – those cities were cut from the data to avoid issues with plotting. Overall, the map data did serve its purpose well, by allowing me to plot values from the cities on a map which gives the user a better understanding of the data.

Chapter 3

Design

3.1 Overview

The main goal of this program, as stated in section 1.2, is mainly to give the user an easy way to learn and understand real estate and rental properties. To achieve this, the program gives the user an easy-to-use interface to visually see and analyze data. Figure 3.1, shown below, shows the initial screen and style of the interface. The program also scores various major cities to give the user a better understanding of real estate. In this chapter, I will go over the various functions I wrote to achieve this.

Figure 3.1: Introduction Screen

3.2 Back-End

In this back-end section, I will go over the various functions I wrote to pull, modify, plot, and analyze data. These back-end functions were all written in their own file,

'GenPriceGraph.py'. This section will be organized by function, in various order, and will only include major functions used. Also, these functions are all written in python with visual studio.

Function init()

This function initializes the global variables used in other functions. This includes means, medians, standard deviations, maxes, and mins. It also reads in the csv files to a global pandas array, to minimize the need for other functions to read in csv files. The function is shown below in figure 3.2.

```
def init():
    global hp, hpMax, hpMin
    global hr, hrMax, hrMin

    hp = pd.read_csv('data/House_Prices_Fixed.csv')
    hr = pd.read_csv('data/House_Rental.csv')

    #get min and maxes
    hpMax = hp.iloc[:,2:].max(axis=0).max()
    hpMin = hp.iloc[:,2:].min(axis=0).min()

    hrMax = hr.iloc[:,2:].max(axis=0).max()
    hrMin = hr.iloc[:,2:].min(axis=0).min()



    # calculating standard deviation across price data
    med = hp.iloc[:,2:].median(axis=1)[0]

    hpStd = hp.copy(deep=True)
    hpStd = hpStd.iloc[:,2:]
    hpStd = hpStd.to_numpy()
    hpStd = np.nan_to_num(hpStd,nan=med)
    hpStd = np.std(hpStd)


    return 0
```

Figure 3.2: init function

Function getRegionData(RegionNum)

This function takes a region number, then uses that region number to pull information on a specific region from a csv file. It is used mainly in plotting functions or other functions that require a specific region be pulled. The function is shown below in figure 3.3.

```
def getRegionData(RegionNum):
    """
    Gets a numpy array, of a specific city, based on Region Number
    """
    hp = pd.read_csv('data/House_Prices.csv')
    hp.drop(hp.iloc[:,1:5], inplace=True, axis=1)
    hp = hp.loc[hp['RegionID'] == RegionNum]
    hp = hp.iloc[:,1:]
    hp=hp.transpose(copy=False)
    hp=hp.to_numpy()

    return hp
```

Figure 3.3: getRegionData function

Function plotRegionData(RegionNum)

This function takes a region number, pulls the specific information about that region from a csv. Then it plots that data via matplotlib. It is used to visually show data plotted versus time. Another function, plotRegionRent is very similar to this function except it pulls from rent information. The plotRegionData function is shown below in figure 3.4.

```
def plotRegionData(RegionNum):
    """
    Plots House Price Data based on the Region Number
    """
    data = hp.loc[hp['RegionID'] == int(RegionNum)].iloc[0]
    reg = data[1]
    data=data[2:]
    data = data.to_numpy()
    y = pd.Series(range(0,data.shape[0]))
    plt.tick_params(left = True, right = False , labelleft = True ,labelbottom = False, bottom = False)
    plt.scatter(y,data,s=1)
    plt.xlabel('Time')
    plt.ylabel('Cost')
    plt.title('Avg House Price in '+str(reg))
    plt.show()

    return 0
```

Figure 3.4: plotRegionData function

Function plotMap()

This function processes the overall score for each region. Then combines that information with the cities latitude and longitude into a pandas dataframe. This dataframe is used in combination of geopandas and matplotlib to plot the values onto a United States map. The function is shown below in figure 3.5, derived from [8].

```
def plotMap():
    """
    Plots a normalized val on to a US map
    """
    coords = pd.read_csv('data/Coords.csv', encoding='ISO-8859-1')

    regions = hp['RegionID']

    mapi = pd.DataFrame(columns=['lat','lng','val'])

    for i in range(len(regions)):
        tmp = coords.loc[coords['RegionID'] == regions[i]]
        if not tmp.empty:
            tmp.iloc[0]
            lat = float(tmp['lat'])
            lng = float(tmp['long'])

            mapi = mapi.append({'lat': lat, 'lng': lng, 'val': int(getScore(regions[i]))}, ignore_index=True)


    fig, ax = plt.subplots(figsize=(20,20))

    countries = gpd.read_file('data/usa-states-census-2014.shp')

    # val = MinMaxScaler().fit_transform(np.array(mapi['val']).reshape(-1,1))
    # siz = ((val+1)*5)**2

    countries.plot(color="lightgray",ax=ax)
    mapi.plot(x="lng",y="lat",kind="scatter",c="val",colormap="YlOrRd",ax=ax,alpha=0.5)
    ax.grid(b=True, alpha=0.5)

    plt.show()

    return 0
```

Figure 3.5: plotMap function

Function getScore(RID)

This function generates the overall score for a specified region. It does this through the weighted combination of four different functions that will be listed below. Those functions are genDif, genCost, genPredRent, and genYTP. The getScore function is shown below in figure 3.6.

```
def getScore(RID):
    """
    Generate score for a region, out of 1000
    """
    # adjustable parameters
    pDif = 0.2
    pCost = 0.1
    pRent = 0.1
    pYTP = 0.6

    Score = pDif*genDif(RID)+pCost*genCost(RID)+pRent*genPredRent(RID)+pYTP*genYTP(RID)

    return Score
```

Figure 3.6: getScore function

Function genDif(RID)

This function generates a normalized score, out of 1000, based on values obtained from the difference between the current cost and the predicted cost. The predicted cost is generated through the ses function described after this one. The genDif function is shown below in figure 3.7.

```
def genDif(RID, grabFlag=0):
    # predicited cost of house - current
    # using forecasting - simple exponential smoothing

    data = hp.loc[hp['RegionID'] == RID].iloc[0]
    data = data[2:]

    test_as = np.arange(0.2,0.9,0.05)
    minA=0.1
    minSSE=ses(data.to_numpy(),1,0.1)

    for i in range(len(test_as)):
        a = test_as[i]
        tmp = ses(data.to_numpy(),1,a)
        if tmp<minSSE:
            minA=a
            minSSE=tmp

    pred = ses(data.to_numpy(),3,minA,grabFlag=1, printFlag=0)


    dif = data['7/31/2022'] - pred

    # get min and max for avg house prices
    mind = -1*13000
    maxd = 13000

    if dif>maxd:
        score=1000
    elif dif<mind:
        score=0
    else:
        score = (dif-mind)/(maxd-mind)
        score *= 1000

    if grabFlag==1:
        return dif

    return score
```

Figure 3.7: genDif function

Function ses(x, n, a)

The ses function, ses stands for simple exponential smoothing. Simple exponential smoothing is a time series forecasting methods. I learned this method this semester while taking MAT 471 Time Series Analysis. The method is used in time models to predict a future value based on previous values in the sequence. It does this by applying a weight to each previous value, the weights decrease the further away the values get from the start. This method allows a pretty accurate value to be predicted and it only needs the sequence. The outline to this method I derived from an online source, cited here [3]. Please see figure 3.8 below.

```
def ses(x, n, a, grabFlag=0, printFlag=0):
    cols = len(x)
    x = np.append(x,[np.nan]*n)

    f = np.full(cols+n,np.nan)
    f[1]=x[0]

    for i in range(2,cols+1):
        f[i] = a*x[i-1]+(1-a)*f[i-1]

    f[cols+1:]=f[i]
    df = pd.DataFrame.from_dict({"Demand":x,"Forecast":f,"Error":x-f})

    RMSE = (df["Error"]**2).sum()/len(df)
    # print("SSE:",round(RMSE,2))
    if printFlag==1:
        df.index.name = "Periods"
        df[["Demand","Forecast"]].plot(title="Simple Smoothing",style=["-","--"])
        plt.show()

    if grabFlag == 0:
        return RMSE
    else:
        return f[cols+1]
```

Figure 3.8: ses function

Function genCost(RID)

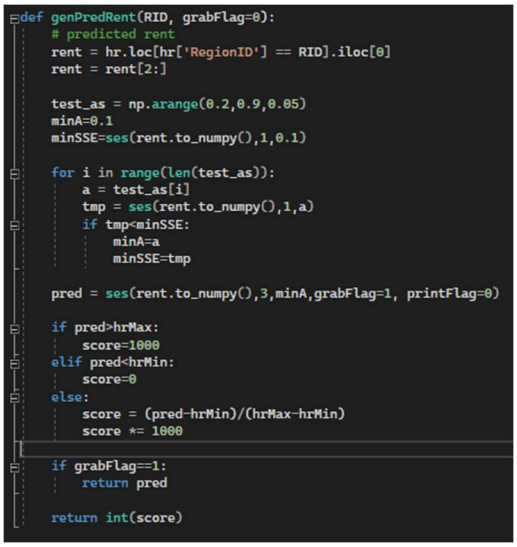
This function generates a score, up to 1000, based on the normalized value of the average price of a house in the city. Cheaper houses receive higher scores, as one would like to see cheaper cities be ranked higher due to the increased ability to purchase more property. See figure 3.9 below.

```
def genCost(RID, grabFlag=0):
    # cost of house
    # normalize then * by 1000
    cost = hp.loc[hp['RegionID'] == RID, '7/31/2022'].iloc[0]
    costTrue=cost
    cost = -1000 * ((cost - hpMin)/(hpMax - hpMin)) + 1000
    cost = math.trunc(cost)

    if grabFlag==1:
        return int(costTrue)

    return int(cost)
```

Figure 3.9: genCost function

Function genPredRent(RID)

This function generates a score, up to 1000, based on the normalized value of rent price predicted into the future. The future rent price is predicted via the ses function, shown in figure 3.8. The scores are ranked based on how high the rent price it. See figure 3.10 below.

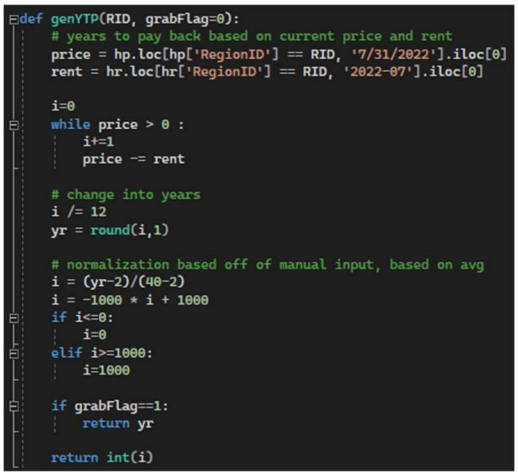


```
def genPredRent(RID, grabFlag=0):
    # predicted rent
    rent = hr.loc[hr['RegionID'] == RID].iloc[0]
    rent = rent[2:]

    test_as = np.arange(0.2,0.9,0.05)
    minA=0.1
    minSSE=ses(rent.to_numpy(),1,0.1)

    for i in range(len(test_as)):
        a = test_as[i]
        tmp = ses(rent.to_numpy(),1,a)
        if tmp<minSSE:
            minA=a
            minSSE=tmp

    pred = ses(rent.to_numpy(),3,minA,grabFlag=1, printFlag=0)

    if pred>hrMax:
        score=1000
    elif pred<hrMin:
        score=0
    else:
        score = (pred-hrMin)/(hrMax-hrMin)
        score *= 1000

    if grabFlag==1:
        return pred

    return int(score)
```

Figure 3.10: genPredRent function

Function genYTP(RID)

This function is weighted the most in the getScore function, due to it holding the most value out of all the other scoring methods. This method predicts how many years it would take, at current house price and predicted rent price, to fully pay off a property. This is most important because it's the fast way for a user to gain a profit on a property. It also helps the user understand the importance of cost vs rent ratios. See figure 3.11 below.



```
def genYTP(RID, grabFlag=0):
    # years to pay back based on current price and rent
    price = hp.loc[hp['RegionID'] == RID, '7/31/2022'].iloc[0]
    rent = hr.loc[hr['RegionID'] == RID, '2022-07'].iloc[0]

    i=0
    while price > 0 :
        i+=1
        price -= rent

    # change into years
    i /= 12
    yr = round(i,1)

    # normalization based off of manual input, based on avg
    i = (yr-2)/(40-2)
    i = -1000 * i + 1000
    if i<=0:
        i=0
    elif i>=1000:
        i=1000

    if grabFlag==1:
        return yr

    return int(i)
```

Figure 3.11: genYTP function

3.3 Front-End

The front-end portion of this program really helps fulfill the goal that the program is easy-to-use. The front-end was written using tkinter, a generic python gui library [4]. It is split into a few screens and some buttons that help plot or show data and scores. Each page is kind of like its own function. Therefore, this section will be split by page. In total the whole program is spanned across four pages and various buttons. Also, all pages were written in the 'main.py' file of the program.

Main Screen

This page serves as a welcome to the program and gives a few buttons that lead to the other pages. The main purpose of this page is to pretty much give the user an easy way to see and access the other pages. The screens are all similar are pretty straightforward. Therefore, I will only show the code for the main screen below. If you would like to see more, please refer to the code on my github listed at the end. See figure 3.12 below.

```
def main():

    g.init()

    root = tk.Tk()

    root.geometry("800x500")
    root.title("Real Estate Predictor")

    label = tk.Label(root, text="Welcome to REP", font=('Terminal', 18))
    label.pack(padx=20,pady=20)

    btn1 = tk.Button(root, text="Overall", font=('Terminal',16), command=overall)
    btn1.place(x=10,y=50)

    btn2 = tk.Button(root, text="Specific", font=('Terminal',16), command=specific)
    btn2.place(x=10,y=100)

    btn3 = tk.Button(root, text="Analyze", font=('Terminal',16), command=analyze)
    btn3.place(x=10,y=150)

    root.mainloop()

    return 0
```

Figure 3.12: Main Screen

Overall Screen

When entering this screen, the program generates the overall scores for all available cities in the background. This is done via the getScore function, shown in figure 3.6. Afterwards, it lists the scores and cities in ascending order – so that the user can easily see the scores. It includes one button located at the top of the screen. This button lets the user plot these scores on a map of the United States.

Specific Screen

This screen functions as a way for the user to see the specific plots of house prices and rental prices in a region over time. It also allows the user to compare plots on the same graph, just by entering an additional region. The screen shows a list of available cities and their region numbers on the left-hand side. On the right side an entry box is there to let the user enter a region number and then plot its corresponding price or rent plot.

Analyze Screen

This screen is used to show the individual scores for each of the four methods used in getScore, figure 3.6. Each button corresponds to one of the functions – genDif, genCost, genPredRent, or genYTP. Once a button is pushed it prints the scores in ascending order in the center of the screen.

Chapter 4

Implementation

4.1 Environment

In this project, I used an anaconda environment. The setup to this environment was very basic – as not a ton of packages were required to accomplish the goals of the project. The packages included pandas, geopandas, numpy, matplotlib, and sklearn. Geopandas and sklearn required a pip install. Pandas is a library for data analysis, it was used for a lot of data reading and manipulation, [7]. Geopandas is a library used to make working with maps easier. Numpy allows a lot of the mathematical functions used in multi-dimensional arrays, the real estate data. Matplotlib is a library that plots data, it is used for all the plotting and graphing done in this project. Sklearn, or Scikit-learn, is a tool used for predictive data analysis. All these libraries together helped form the project.

4.2 User Guide

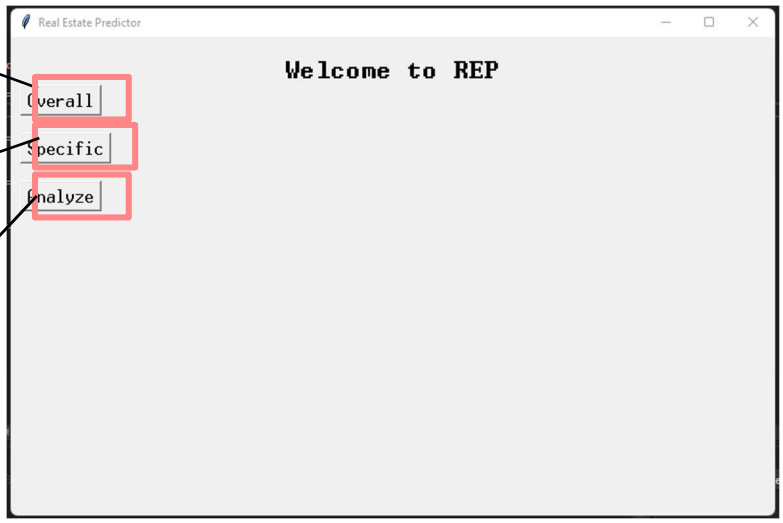
This section will give the user a brief and informative overview on how to use the program. This will include information on each screen and button.

Main Screen

Send to Overall Screen

Send to Specific Screen

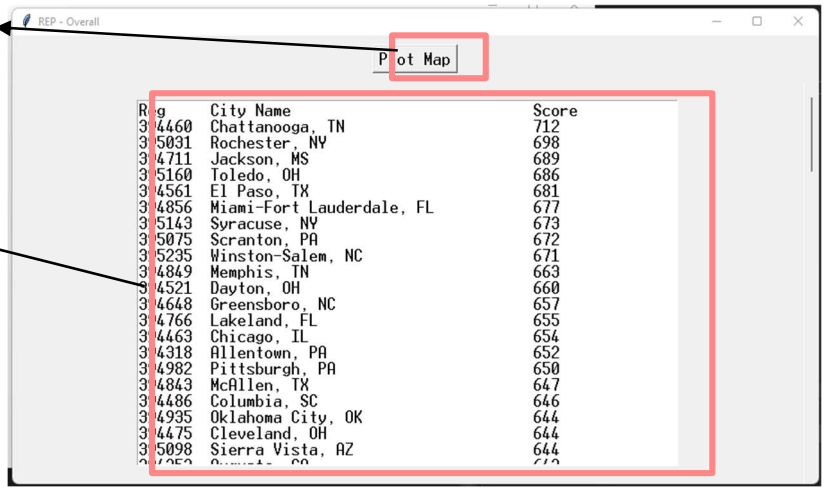
Send to Analyze Screen



Overall Screen

Plot a map of the values below

List of cities, ranking by score, based on overall score

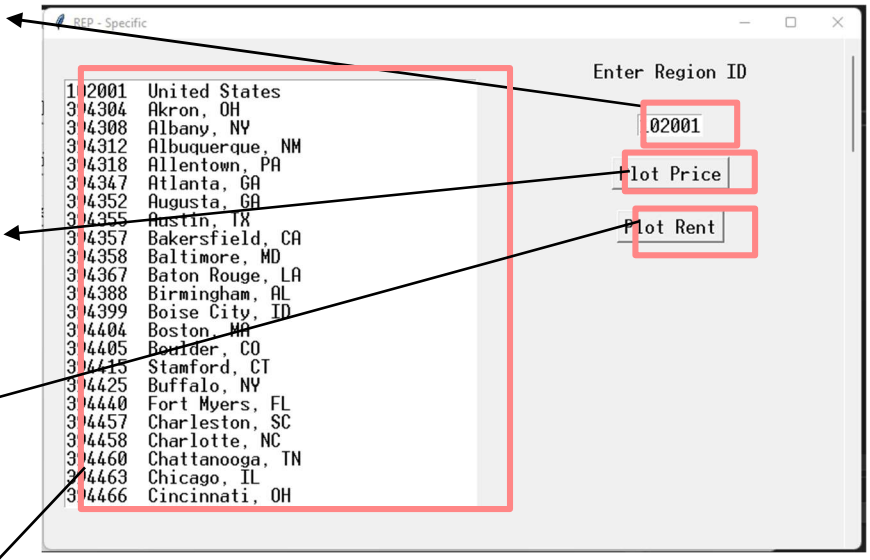


Specific Screen

Entry box for region number

Plot the price plot of whatever is in the entry box

Plot the rent plot of whatever is in the entry box



Displays a list of all available cities and their respective region numbers

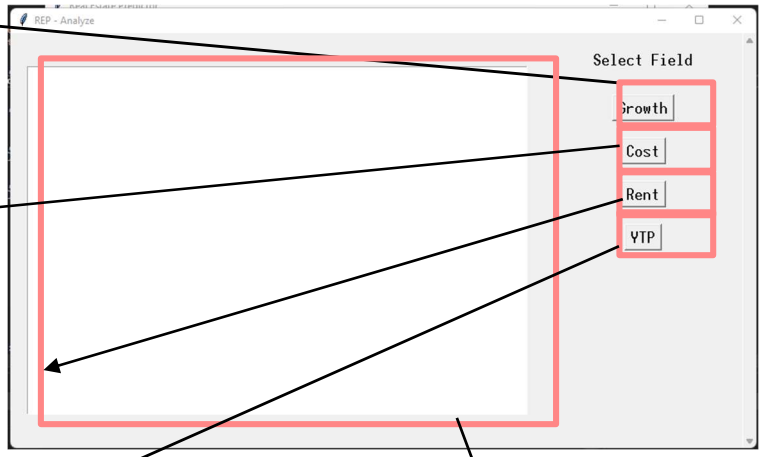
Analyze Screen

Shows scores on growth, genDif

Shows scores on cost, genCost

Shown scores on rent, genPredRent

Shows scores on YTP, genYTP



Displays the cities and scores

Chapter 5

Conclusion

5.1 Accomplishments

Throughout this project, I believe I have accomplished my main goal as well as some additional milestones. The interface and background processes I built all serve to give the user a better understanding of real estate and real estate trends. The programs' ability to display and rank data based on various factors lets the user analyze nation real estate trends. My other goal, of showing the most optimal area to buy rental properties was also fulfilled. This was done by the overall functionality that ranked cities based on their respective scores.

5.2 Future Work

I definitely learned a ton over the course of this project. The first and foremost being the importance of data. Not just having a bunch of data – but organizing the data in a way that makes using it easier. In the future, if I was presented with a similar project, I would probably use a better way to store data instead of csv's. There could be a small database that stores the information that would allow an easier and faster way to access and process the data. All in all, I enjoyed this project and learned a lot about data analysis and interfaces.

Appendix

REP	Real estate predictor, this references the main program
Front-End	The part of the application the user interacts with directly, User interface
Back-End	The part of the application that runs processes in the background
Rental Property	A property that primarily functions as a space to be rented
Data	Information collected and stored in a single place
Score	A measurement used to compare various information
YTP	Years to pay back, as in how long it would take to repay fully
CSV	Comma-separated file, a text file that stores information that's separated by commas
Function	A section of code that takes an input and gives an output
Zillow	A platform used to buy or sell or rent properties

References

[1] GeeksforGeeks. “GeeksforGeeks | a Computer Science Portal for Geeks.” *GeeksforGeeks*, www.geeksforgeeks.org.

[2] Jc. “GeoPandas Tutorial: How to Plot US Maps in Python.” *jcutrer.com*, 29 July 2020, jcutrer.com/python/learn-geopandas-plotting-usmaps.

[3] Pierre, Sadrach. “A Guide to Time Series Forecasting in Python.” *Built In*, 6 Oct. 2021, builtin.com/data-science/time-series-forecasting-python.

[4] “The Python Standard Library.” *Python Documentation*, docs.python.org/3/library.

[5] Terrazas, Aaron. “Housing Data.” *Zillow Research*, 8 Nov. 2022, www.zillow.com/research/data.

[6] *US Cities Database | Simplemaps.com*. simplemaps.com/data/us-cities.

[7] *User Guide — Pandas 1.5.2 Documentation*. pandas.pydata.org/docs/user_guide/index.html.

[8] Yosovzon, Udi. “The Easiest Way to Plot Data From Pandas on a World Map.” *Medium*, 30 Dec. 2021, towardsdatascience.com/the-easiest-way-to-plot-data-from-pandas-on-a-world-map-1a62962a27f3.