SEAS 6414

Spring 2024

Assignment 5, Michael Wacey

Dr. Adewale Akinfaderin

For an easier week, that was a lot of work. Interestingly, I tried Google Gemini(Bard) for some exercises. But that did not help because things were different enough. It may have given me a starting point but there was still a lot to do.

I have embedded the graphs in this file. They are numbered the same as the problem with an A for the first one and an B for the second one, if there is a second one.

Let me know if you want to see the source file. This is the executed file and has everything in it. But I am happy to share the source file. This assignment is in GitHub at <https://github.com/OwlSaver/GWU>.

# Execution

#######################################################################################

# Week 5

#######################################################################################

#######################################################################################

# Problem 1 - Data Cleaning and Exploration

#######################################################################################

Problem:

- Load the zillow feature sample.csv dataset using Pandas and report any missing values

per column. Create a strategy to handle these missing values, justifying your approach.

- Generate a summary table that shows the mean, median, and standard deviation of

taxvaluedollarcnt, structuretaxvaluedollarcnt, and landtaxvaluedollarcnt

for properties built in each decade (1960s, 1970s, etc.).

Code:

import numpy as np

import pandas as pd

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

print(f"The Zillow Feature Sample has {ZillowFeatureSample.isnull().sum().sum()} missing values.")

print(f"The Zillow Feature Sample missing values by feature:")

print(ZillowFeatureSample.isnull().sum())

print("")

print("My strategy to address the missing values is based on the number of missing features and")

print("an understanding of the data. The steps I would go through:")

print(" 1) For each feature, determine if there is a logical value for it, if it is missing. For")

print(" example, if the number of fireplaces is missing, then assume it is zero.")

print(" 2) Any feature that is missing under 2% of the values and is continuous would be")

print(" replaced with the mean od the non missing values.")

print(" 3) Any feature that is missing more than 60% of its values, would be removed.")

print(" 4) Next I would sort the rows by the number of missing features they have. I would look")

print(" for a natural break where there are two many missing features. These rows would be")

print(" removed.")

print(" 5) I would then compare the mean and standard deviation of each feature in the original")

print(" dataset to the one adjusted for missing values. If any large differences appear, I")

print(" look into why this happened and what can be done about it.")

print("")

ZillowFeatureSample['decadebuilt'] = ["0000" if yb != yb else str(int(yb-(yb%10))) for yb in ZillowFeatureSample.yearbuilt]

ZillowFeatureSampleSummary = ZillowFeatureSample.groupby("decadebuilt").agg(

row\_count=("taxvaluedollarcnt", "count")

, taxvaluedollarcnt\_mean=("taxvaluedollarcnt", "mean")

, taxvaluedollarcnt\_median=("taxvaluedollarcnt", "median")

, taxvaluedollarcnt\_std=("taxvaluedollarcnt", "std")

, structuretaxvaluedollarcnt\_mean=("structuretaxvaluedollarcnt", "mean")

, structuretaxvaluedollarcnt\_median=("structuretaxvaluedollarcnt", "median")

, structuretaxvaluedollarcnt\_std=("structuretaxvaluedollarcnt", "std")

, landtaxvaluedollarcnt\_mean=("landtaxvaluedollarcnt", "mean")

, landtaxvaluedollarcnt\_median=("landtaxvaluedollarcnt", "median")

, landtaxvaluedollarcnt\_std=("landtaxvaluedollarcnt", "std")

)

print("")

print("Summary table:")

print(ZillowFeatureSampleSummary)

Execution:

The Zillow Feature Sample has 284030 missing values.

The Zillow Feature Sample missing values by feature:

parcelid 0

airconditioningtypeid 7219

architecturalstyletypeid 9987

basementsqft 9996

bathroomcnt 13

bedroomcnt 13

buildingclasstypeid 9961

buildingqualitytypeid 3530

calculatedbathnbr 388

decktypeid 9932

finishedfloor1squarefeet 9305

calculatedfinishedsquarefeet 149

finishedsquarefeet12 859

finishedsquarefeet13 9974

finishedsquarefeet15 9388

finishedsquarefeet50 9305

finishedsquarefeet6 9928

fips 13

fireplacecnt 8953

fullbathcnt 388

garagecarcnt 6978

garagetotalsqft 6978

hashottuborspa 9827

heatingorsystemtypeid 3757

latitude 13

longitude 13

lotsizesquarefeet 925

poolcnt 8162

poolsizesum 9894

pooltypeid10 9937

pooltypeid2 9890

pooltypeid7 8275

propertycountylandusecode 14

propertylandusetypeid 13

propertyzoningdesc 3411

rawcensustractandblock 13

regionidcity 210

regionidcounty 13

regionidneighborhood 6078

regionidzip 42

roomcnt 13

storytypeid 9996

threequarterbathnbr 8929

typeconstructiontypeid 9980

unitcnt 3400

yardbuildingsqft17 9746

yardbuildingsqft26 9988

yearbuilt 166

numberofstories 7655

fireplaceflag 9989

structuretaxvaluedollarcnt 144

taxvaluedollarcnt 119

assessmentyear 13

landtaxvaluedollarcnt 210

taxamount 66

taxdelinquencyflag 9816

taxdelinquencyyear 9816

censustractandblock 240

dtype: int64

My strategy to address the missing values is based on the number of missing features and

an understanding of the data. The steps I would go through:

1) For each feature, determine if there is a logical value for it, if it is missing. For

example, if the number of fireplaces is missing, then assume it is zero.

2) Any feature that is missing under 2% of the values and is continuous would be

replaced with the mean od the non missing values.

3) Any feature that is missing more than 60% of its values, would be removed.

4) Next I would sort the rows by the number of missing features they have. I would look

for a natural break where there are two many missing features. These rows would be

removed.

5) I would then compare the mean and standard deviation of each feature in the original

dataset to the one adjusted for missing values. If any large differences appear, I

look into why this happened and what can be done about it.

Summary table:

row\_count taxvaluedollarcnt\_mean taxvaluedollarcnt\_median taxvaluedollarcnt\_std structuretaxvaluedollarcnt\_mean structuretaxvaluedollarcnt\_median structuretaxvaluedollarcnt\_std landtaxvaluedollarcnt\_mean landtaxvaluedollarcnt\_median landtaxvaluedollarcnt\_std

decadebuilt

0000 148 282,547.80 40,953.50 925,541.24 158,973.64 24,109.50 508,813.27 306,692.13 119,441.50 669,489.95

1860 1 2,600,588.00 2,600,588.00 NaN 104,023.00 104,023.00 NaN 2,496,565.00 2,496,565.00 NaN

1880 4 473,815.25 450,008.00 234,392.08 141,027.25 128,322.00 83,936.50 332,788.00 283,388.50 226,589.90

1890 10 298,284.10 195,041.50 210,090.87 100,655.40 38,199.50 110,791.41 197,628.70 152,385.00 136,610.95

1900 103 294,758.97 254,328.00 232,803.52 114,436.28 75,057.00 112,924.13 180,322.69 146,301.00 148,736.98

1910 158 397,018.52 252,592.00 724,096.14 124,393.66 73,706.00 197,842.06 272,624.86 166,412.00 553,421.01

1920 673 430,727.81 279,054.00 636,701.47 135,394.68 90,315.00 185,213.51 295,534.31 172,972.00 483,498.19

1930 349 499,617.99 336,282.00 592,583.83 174,301.76 105,911.00 236,973.07 325,316.23 218,224.00 398,038.63

1940 980 366,712.86 290,323.50 333,351.21 124,400.49 96,000.00 109,412.82 242,439.30 182,391.00 252,038.49

1950 2049 357,176.30 279,139.00 459,852.41 119,710.93 94,500.00 110,366.36 237,582.22 168,412.00 393,884.50

1960 1372 395,530.76 324,814.00 390,546.73 146,786.26 118,833.00 134,046.67 248,925.93 176,199.00 301,753.31

1970 1389 391,092.63 316,000.00 318,708.58 155,448.88 130,979.00 97,463.30 235,925.52 160,378.00 258,126.44

1980 1210 463,558.15 342,899.50 776,381.84 215,180.24 166,615.50 252,129.08 249,201.72 158,589.50 548,221.89

1990 618 584,373.33 460,059.00 530,667.33 279,759.40 217,237.50 223,138.51 306,099.85 213,829.00 358,105.63

2000 660 787,722.41 572,573.00 777,851.88 385,706.62 280,808.50 376,648.70 403,237.73 272,929.50 486,368.40

2010 157 985,455.25 696,385.00 1,161,067.39 455,226.42 336,717.00 466,447.01 537,070.49 374,656.00 836,742.84

#######################################################################################

# Problem 2 - Feature Engineering

#######################################################################################

Problem:

- Create a new feature Age that represents the age of each property from the yearbuilt

column, considering the dataset's latest assessmentyear.

- Develop a binary feature HasPool based on the poolcnt column, where 1 indicates

the presence of a pool and 0 or NaN indicates no pool.

- Calculate and return the descriptive statistics for the age of the properties. Specifically,

report the median age of the properties based on the yearbuilt and the

latest assessmentyear.

- Generate and plot a bar chart of the counts of the binary feature HasPool created

earlier. Set the y-axis to a logarithmic scale to better visualize the distribution of

properties with and without pools.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSample['Age'] = [ZillowFeatureSample["assessmentyear"].max() - yb for yb in ZillowFeatureSample.yearbuilt]

ZillowFeatureSample['HasPool'] = [1 if pc > 0 else 0 for pc in ZillowFeatureSample.poolcnt]

print("")

print("Summary table:")

print(ZillowFeatureSample[["Age", "yearbuilt", "poolcnt", "HasPool"]])

print("")

print(f"The medan age of the properties based on Year Built and latest assement year is {ZillowFeatureSample["Age"].median()}.")

print("")

x = ZillowFeatureSample['HasPool'].value\_counts().plot(kind='bar')

x.set\_yscale('log')

plt.title(f"Problem {PNumber} A:Houses with and without pools")

plt.show()

Execution:

Summary table:

Age yearbuilt poolcnt HasPool

0 61.00 1,955.00 NaN 0

1 4.00 2,012.00 NaN 0

2 59.00 1,957.00 NaN 0

3 10.00 2,006.00 NaN 0

4 29.00 1,987.00 NaN 0

... ... ... ... ...

9995 16.00 2,000.00 1.00 1

9996 59.00 1,957.00 NaN 0

9997 58.00 1,958.00 NaN 0

9998 37.00 1,979.00 1.00 1

9999 39.00 1,977.00 NaN 0

[10000 rows x 4 columns]

The medan age of the properties based on Year Built and latest assement year is 53.0.

A screenshot of a computer

Description automatically generated

#######################################################################################

# Problem 3 - Correlation Analysis

#######################################################################################

Problem:

- Using NumPy, calculate the Pearson correlation coefficient between bedroomcnt

and bathroomcnt. Visualize the correlation matrix of the numerical features of the

dataset using a heatmap in matplotlib.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSampleSmall = ZillowFeatureSample.loc[:,['bedroomcnt','bathroomcnt']]

print(ZillowFeatureSampleSmall.corr(numeric\_only=True))

plt.imshow(ZillowFeatureSampleSmall.corr(numeric\_only=True), cmap='hot', interpolation='nearest')

plt.title(f"Problem {PNumber} A: Correlation of Bedrooms and Bathrooms")

plt.show()

Execution:

bedroomcnt bathroomcnt

bedroomcnt 1.00 0.70

bathroomcnt 0.70 1.00

Graphical user interface, text

Description automatically generated

#######################################################################################

# Problem 4 - Geospatial Analysis

#######################################################################################

Problem:

- Plot a scatter plot of latitude and longitude to visualize the geographical distribution

of properties. Overlay this plot with a density estimate to highlight property

clusters.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from scipy.stats import gaussian\_kde

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

print("I did the scatter and looked for a density plot to overlay. I could not find one. So,")

print("instead I have calculated an area and used it to color the plots to show density. But")

print("the density seems to be consistent. So, it does not really show much.")

# Get just the columns that we need and drop any rows with NaN. The gaussian calculation

# does not work with NaN values.

ZillowFeatureSampleSmall = ZillowFeatureSample.loc[:,['longitude','latitude']].dropna()

z = gaussian\_kde(ZillowFeatureSampleSmall["longitude"])(ZillowFeatureSampleSmall["latitude"])

fig, ax = plt.subplots()

ax.scatter(ZillowFeatureSampleSmall["longitude"], ZillowFeatureSampleSmall["latitude"], c=z, s=100)

plt.title(f"Problem {PNumber} A: Scatter Plot")

plt.show()

Execution:

I did the scatter and looked for a density plot to overlay. I could not find one. So,

instead I have calculated an area and used it to color the plots to show density. But

the density seems to be consistent. So, it does not really show much.

Text

Description automatically generated

#######################################################################################

# Problem 5 - Market Value Analysis

#######################################################################################

Problem:

- Visualize the trend of average taxvaluedollarcnt over the years using a line chart.

Add a shaded area representing the 95% confidence interval for the average values.

- Create a boxplot to compare the distribution of taxvaluedollarcnt across different

buildingqualitytypeid.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSample['assessyear'] = ["0000" if ay != ay else str(int(ay)) for ay in ZillowFeatureSample.assessmentyear]

ZillowFeatureSampleSummary = ZillowFeatureSample.groupby("assessyear").agg(

taxvaluedollarcnt\_mean=("taxvaluedollarcnt", "mean")

, taxvaluedollarcnt\_std=("taxvaluedollarcnt", "std")

).dropna()

ZillowFeatureSampleSummary['plus95'] = ZillowFeatureSampleSummary.apply(lambda row: row.taxvaluedollarcnt\_mean + (2 \* row.taxvaluedollarcnt\_std), axis=1)

ZillowFeatureSampleSummary['minus95'] = ZillowFeatureSampleSummary.apply(lambda row: row.taxvaluedollarcnt\_mean - (2 \* row.taxvaluedollarcnt\_std), axis=1)

plt.plot(ZillowFeatureSampleSummary.index, ZillowFeatureSampleSummary["taxvaluedollarcnt\_mean"], label="Average")

plt.fill\_between(ZillowFeatureSampleSummary.index, ZillowFeatureSampleSummary["plus95"], ZillowFeatureSampleSummary["minus95"], alpha=0.2, label="Confidence Interval")

plt.title(f"Problem {PNumber} A: Average with Confidence Interval")

plt.show()

ax = ZillowFeatureSample.boxplot(column="taxvaluedollarcnt", by="buildingqualitytypeid")

plt.title(f"Problem {PNumber} B: Box Plot")

plt.show()

Execution:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

#######################################################################################

# Problem 6 - Tax Analysis

#######################################################################################

Problem:

- Analyze the relationship between taxamount and taxvaluedollarcnt using a scatter plot

and fit a linear regression line to it. Calculate the R-squared value for this

fit.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSampleSmall = ZillowFeatureSample.loc[:,['taxamount','taxvaluedollarcnt']].dropna()

X = np.array(ZillowFeatureSampleSmall["taxvaluedollarcnt"]).reshape(-1, 1)

y = np.array(ZillowFeatureSampleSmall["taxamount"])

# fit the model

reg = LinearRegression().fit(X, y)

ZillowFeatureSampleSmall['predicted'] = reg.predict(X)

rsquare = reg.score(X, y)

fig, ax = plt.subplots()

ax.text(1, 150000, f"$R^2$ = {rsquare}", bbox=dict(facecolor='red', alpha=0.5))

ZillowFeatureSampleSmall.plot.scatter(x = 'taxvaluedollarcnt', y = 'taxamount', ax = ax)

ZillowFeatureSampleSmall.plot.line(x = 'taxvaluedollarcnt', y = 'predicted', color = 'red', ax = ax)

plt.title(f"Problem {PNumber} A: Linear Regression")

plt.show()

Execution:

A screenshot of a computer

Description automatically generated

#######################################################################################

# Problem 7 - Comparative Analysis

#######################################################################################

Problem:

- For properties with numberofstories more than 1, compare the average

calculatedfinishedsquarefeet against those with only 1 story using a bar chart.

- Compare the taxvaluedollarcnt for properties with and without a fireplace (fireplaceflag)

using a violin plot.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSample['storygroup'] = ["Multi" if nos > 1 else "One" for nos in ZillowFeatureSample.numberofstories]

ZillowFeatureSample['fpgroup'] = ["None" if fpf != fpf else "Has" for fpf in ZillowFeatureSample.fireplaceflag]

ZillowFeatureSampleSG = ZillowFeatureSample.groupby("storygroup").agg(

calculatedfinishedsquarefeet\_mean=("calculatedfinishedsquarefeet", "mean")

).dropna()

x = ZillowFeatureSampleSG.plot.bar()

plt.title(f"Problem {PNumber} A: Houses average square feet for one or multi stories")

plt.show()

sb.violinplot(x = 'fpgroup', y = "taxvaluedollarcnt", data = ZillowFeatureSample, inner="stick")

plt.title(f"Problem {PNumber} B: Fireplace Comparison")

plt.show()

Execution:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

#######################################################################################

# Problem 8 - Time-Series Forecasting (Advanced)

#######################################################################################

Problem:

- Group the data by yearbuilt and calculate the annual mean of landtaxvaluedollarcnt.

Using this time series data, create a forecast plot for the next 10 years with a rolling

mean and standard deviation.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.linear\_model import LinearRegression

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

print("")

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

# Drop outliers before 1900

ZillowFeatureSampleYB = ZillowFeatureSample.loc[ZillowFeatureSample['yearbuilt'] > 1900]

ZillowFeatureSampleYB = ZillowFeatureSampleYB.groupby("yearbuilt").agg(

landtaxvaluedollarcnt\_mean=("landtaxvaluedollarcnt", "mean")

).dropna()

print("I searched the web and found several articles about ARIMA / SARIMA as a way to train")

print("Models for prediction. I was not sure that this was the path to take. So, instead")

print("just used regression on a ten year window and predicted the next year from that.")

ZillowFeatureSampleSmall = ZillowFeatureSample.loc[:,['landtaxvaluedollarcnt','yearbuilt']].dropna()

X = np.array(ZillowFeatureSampleSmall["yearbuilt"]).reshape(-1, 1)

y = np.array(ZillowFeatureSampleSmall["landtaxvaluedollarcnt"])

# fit the model

reg = LinearRegression().fit(X, y)

ZillowFeatureSampleSmall['predicted'] = reg.predict(X)

future\_years = np.array(range(2015, 2024))

predicted\_prices = np.array(reg.predict(future\_years.reshape(-1, 1)))

predicted\_data = pd.DataFrame({'yearbuilt': future\_years, 'landtaxvaluedollarcnt':predicted\_prices}).set\_index('yearbuilt')

plt.plot(ZillowFeatureSampleYB, color = "black", label = "History")

plt.plot(predicted\_data, color = "red", label = "Prediction")

plt.ylabel('House Price')

plt.xlabel('Year Built')

plt.legend()

plt.title(f"Problem {PNumber} A: Prediction")

plt.show()

print("OK, that did not work well. So, back to the drawing board. Below is another attempt that I think does")

print("what you asked.")

# Define window size for the rolling window

window\_size = 10

# Calculate rolling mean

ZillowFeatureSampleYB["rolling\_mean"] = ZillowFeatureSampleYB["landtaxvaluedollarcnt\_mean"].rolling(window=window\_size).mean()

# Calculate rolling standard deviation

ZillowFeatureSampleYB["rolling\_std"] = ZillowFeatureSampleYB["landtaxvaluedollarcnt\_mean"].rolling(window=window\_size).std()

# Extend index for 10 years

future\_years = np.array(range(2015, 2024))

# Extend the existing data with NaNs for future dates

ZillowFeatureSampleYB\_extended = pd.concat([ZillowFeatureSampleYB,pd.DataFrame(index=future\_years)])

# Fill NaN values with the last rolling mean

ZillowFeatureSampleYB\_extended["rolling\_mean"] = ZillowFeatureSampleYB\_extended["rolling\_mean"].fillna(method="ffill")

# Calculate the upper and lower bounds based on rolling mean and standard deviation

ZillowFeatureSampleYB\_extended["upper\_bound"] = ZillowFeatureSampleYB\_extended["rolling\_mean"] + 2 \* ZillowFeatureSampleYB\_extended["rolling\_std"]

ZillowFeatureSampleYB\_extended["lower\_bound"] = ZillowFeatureSampleYB\_extended["rolling\_mean"] - 2 \* ZillowFeatureSampleYB\_extended["rolling\_std"]

# Plot observed data, rolling mean, and bounds

plt.figure(figsize=(12, 6))

plt.plot(ZillowFeatureSampleYB.index, ZillowFeatureSampleYB["landtaxvaluedollarcnt\_mean"], label="Observed")

plt.plot(ZillowFeatureSampleYB\_extended.index, ZillowFeatureSampleYB\_extended["rolling\_mean"], label="Rolling Mean")

plt.fill\_between(ZillowFeatureSampleYB\_extended.index, ZillowFeatureSampleYB\_extended["upper\_bound"], ZillowFeatureSampleYB\_extended["lower\_bound"], alpha=0.2, label="Confidence Interval")

# Add labels and title

plt.xlabel("Year")

plt.ylabel("Land Tax Value")

plt.title(f"Problem {PNumber} B: Forecast with Rolling Mean and Standard Deviation")

# Rotate x-axis labels for better readability

plt.xticks(rotation=45)

# Show the plot

plt.legend()

plt.grid()

plt.tight\_layout()

plt.show()

Execution:

I searched the web and found several articles about ARIMA / SARIMA as a way to train

Models for prediction. I was not sure that this was the path to take. So, instead

just used regression on a ten year window and predicted the next year from that.

A screenshot of a computer

Description automatically generated

OK, that did not work well. So, back to the drawing board. Below is another attempt that I think does

what you asked.

A screenshot of a computer

Description automatically generated with medium confidence

#######################################################################################

# Problem 9 - Amenities Impact Analysis

#######################################################################################

Problem:

- Determine how the presence of a hot tub or spa (hashottuborspa) and air conditioning (airconditioningtypeid)

impacts the taxvaluedollarcnt. Use a grouped bar chart to represent the average taxvaluedollarcnt for

properties with and without these amenities.

- Investigate if there is a significant difference in the calculatedfinishedsquarefeet for properties with a

basement (basementsqft) versus those without. Perform a hypothesis test and visualize the results using a

histogram overlaid with the probability density function.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import scipy.stats as stats

import seaborn as sns

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSample['HasHTS'] = ["No Hot Tub" if hhts != hhts else "Hot Tub" for hhts in ZillowFeatureSample.hashottuborspa]

ZillowFeatureSample['HasAC'] = ["No AC" if act != act or act == 5.0 else "AC" for act in ZillowFeatureSample.airconditioningtypeid]

ZillowFeatureSampleAM = ZillowFeatureSample.groupby(["HasHTS", "HasAC"]).agg(

taxvaluedollarcnt\_mean=("taxvaluedollarcnt", "mean")

).dropna()

x = ZillowFeatureSampleAM.plot.bar()

#x.ticklabel\_format(style='plain')

plt.xlabel("Amenities")

plt.ylabel("Average Tax Value ($)")

plt.title(f"Problem {PNumber} A: Average Tax Value based on Amenities")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

ZillowFeatureSampleSF = ZillowFeatureSample.loc[:,['calculatedfinishedsquarefeet','basementsqft']]

ZillowFeatureSampleSF['HasBasement'] = ["No Basement" if bsf != bsf or bsf <= 0 else "Basement" for bsf in ZillowFeatureSampleSF.basementsqft]

ZillowFeatureSampleHasB = ZillowFeatureSampleSF.loc[ZillowFeatureSampleSF['HasBasement'] == "Basement"]

ZillowFeatureSampleHasB = ZillowFeatureSampleHasB.drop(columns=['basementsqft','HasBasement'])

ZillowFeatureSampleNoB = ZillowFeatureSampleSF.loc[ZillowFeatureSampleSF['HasBasement'] != "Basement"]

ZillowFeatureSampleNoB = ZillowFeatureSampleNoB.drop(columns=['basementsqft','HasBasement'])

# Define the null hypothesis

H0 = "Properties with a basement will have more square feet than those without."

# Define the alternative hypothesis

H1 = "Properties with a basement will have the same or fewer square feet than those without."

# Calculate the test statistic

t\_stat, p\_value = stats.ttest\_ind(ZillowFeatureSampleHasB, ZillowFeatureSampleNoB,nan\_policy='omit')

# Print the results

print("Test statistic:", t\_stat)

print("p-value:", p\_value)

# Conclusion

if p\_value != p\_value:

print("t Test failed.")

elif p\_value < 0.05:

print(f"Reject the null hypothesis of {H0}.")

else:

print(f"Failed to reject the null hypothesis of {H0}.")

ZillowFeatureSampleSFA = ZillowFeatureSampleSF.drop(columns=['HasBasement'])

sns.displot(ZillowFeatureSampleSFA, kde=True)

plt.title(f"Problem {PNumber} B: Histogram overlaid with the probability density function")

plt.show()

Execution:

Test statistic: [1.15127171]

p-value: [0.24964842]

Failed to reject the null hypothesis of Properties with a basement will have more square feet than those without..

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

#######################################################################################

# Problem 10 - Neighborhood and Regional Analysis

#######################################################################################

Problem:

- Group the properties by regionidneighborhood and plot a horizontal bar chart

showing the top 10 neighborhoods with the highest average taxvaluedollarcnt.

- Using regionidzip, create a pie chart to display the proportion of total taxamount

contributed by the top 5 zip codes. Include a separate 'other' slice for the remaining

zip codes.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import scipy.stats as stats

pd.options.display.float\_format = '{:,.2f}'.format

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 4000)

ZillowFeatureSample = pd.read\_csv("./gwu/SEAS 6414/zillow\_feature\_sample(1).csv")

ZillowFeatureSampleAM = ZillowFeatureSample.groupby(["regionidneighborhood"]).agg(

taxvaluedollarcnt\_mean=("taxvaluedollarcnt", "mean")

)

ZillowFeatureSampleAM = ZillowFeatureSampleAM.sort\_values('taxvaluedollarcnt\_mean', ascending=False).head(10)

ZillowFeatureSampleAM.plot.barh()

plt.ylabel('Neighborhood')

plt.xlabel('Average Tax Value ($)')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.legend()

plt.title(f"Problem {PNumber} A: Top 10 Neigborhoods")

plt.show()

ZillowFeatureSampleZIP = ZillowFeatureSample.groupby(["regionidzip"]).agg(

taxamount\_sum=("taxamount", "sum")

)

ZillowFeatureSampleZipSort = ZillowFeatureSampleZIP.sort\_values('taxamount\_sum', ascending=False)

ZillowFeatureSampleZipTop5 = ZillowFeatureSampleZipSort.head(5).copy()

# We have to recalulate the mean, since we cannot take the mean of the mean - originally I did mean, sum is simpler

# but this works for mean witht he proper changes and so I am leaving it in the more complicated form.

ZillowFeatureSampleZipFull = ZillowFeatureSample.join(ZillowFeatureSampleZipTop5,on='regionidzip',how='outer',)

ZillowFeatureSampleZipFull['zipgroup'] = ZillowFeatureSampleZipFull.apply(lambda row: 'other' if row.taxamount\_sum != row.taxamount\_sum else row.regionidzip, axis=1)

ZillowFeatureSampleZipSum = ZillowFeatureSampleZipFull.groupby(["zipgroup"]).agg(

taxamount\_sum=("taxamount", "sum")

)

ZillowFeatureSampleZipSum.plot.pie(y='taxamount\_sum',legend=None)

plt.tight\_layout()

plt.ylabel('Total Tax')

plt.title(f"Problem {PNumber} B: Top 5 zip codes")

plt.show()

Execution:

A screenshot of a computer

Description automatically generated

Graphical user interface, text

Description automatically generated