**Randy Garcia**

rgarci10@syr.edu

SUID: 342851174

IST 782MS ADS Portfolio



**Table of Contents**

1. Professional Resume
2. Overview
3. Demonstrate data collection and effective structuring.
   1. Conceptual example that demonstrates data collection
   2. Conceptual example that demonstrates data structuring
4. Demonstrate pattern identification and data visualization.
   1. Predictive model project
   2. Visualizations of distribution and charts
5. Analyze data and create recommendations based on findings.
   1. Conceptual example to piece together predictive pricing for vehicle resale values
   2. Use web scraping strategy to pull stock data and perform Stock Portfolio optimization



Project Overview

I am currently a Financial Planning and Analysis Lead Analyst at a major bank. I work with substantial amounts of financial data related to consumer deposits, varying from deposit product balances, sales numbers, transactions, number of households, and other financial statement related information. My key work duties include accessing different data source systems in order to put together historical and forward looking analysis reports, presentations with graphs and charts, and many ad hoc analysis requested by various stakeholders that require me to take data and put it in a readable and organized form to provide insight and aid in management decision making. Although most data sources provide structured data, I often times need to combine data from various databases and data warehouses. Depending on the business and management needs, I have to be able to pull data then manipulate it into usable reports that can be used to create a story of bank’s financials. I often pull data directly management reporting systems and then then disseminate them to various audiences.

When I enrolled in the Applied Data Science master's program, my goal was to acquire new technical skills and become more marketable as an employee, advance my career, become a more valuable and productive business asset, and gain skills that I can apply in my professional everyday life.

Throughout my time in the Applied Data Science master’s program, I have been able to enhance both my technical skills and my people skills. Below is a list of skills that I have been able to acquire and refine during the program:

* Develop stronger presentation and communication skills conveying data and analyses to various individuals; like senior executives, risk managers, technology specialists, while being able to discern what level of information they need to receive.
* Learn best practices in data science which helped me organize my ideas and programming to create effective and easy to understand models and analyses that can be shared with peers and colleagues.
* Gather and structure data effectively.
* Recognize patterns in data through visualization, statistical analysis, and data mining.
* Formulate alternative strategies based on the analyzed data.

**Demonstrate data collection and structuring effectively**

Conceptual works that demonstrates data collection

* R
* Using R programming I looked at various processed and cleaned data for analytical use:
  + - * Reading data from a csv file into a data-frame
      * Summarize the ingested data.
      * Establish if data is missing and remove.
      * Create sub data frames.
      * Structure data into desired format.
* Upon Extraction and preprocessing, I examine the datasets and determine patterns by:
  + - * Assessing the data types of variables
      * Utilizing Association rule mining, and Clustering to discover readable relations between variables.
* Create visuals that display data stories by:
  + - * Choosing the appropriate type of visualization for datasets.
      * Create conceptual data plot graphs.

**IST 687 Final Project**

**Introduction**

This report presents the findings of the Top 1,000 Most Popular HULU Shows dataset consisting of 1000 rows and 181 columns consisting of various aspects of the HULU shows, such as episode count, network, and genre. The objective of this report is to look at the variables that affect a shows popularity to provide insight to HULU on what changes they can make to less popular shows to make them popular, and thus increase business. Our team developed various business questions to steer our data analysis direction. After reviewing and reading in the data, our team cleaned, munged, and prepared the data set to create data visualization plots to better illustrate our findings. Our team also developed models to analyze the data to identify attributes of various variables that have the most impact on show popularity.

**Business Questions:**

1. What makes a show “popular”? What factors make a show “popular”?
2. How does show popularity compare to other values within the dataset?
3. Are there correlations between variables that result in a show being “popular”?

**Data Cleanse/Munge/Preparation**

1. We began by reading in the data and linking it to a data frame name.
2. Due to the original dataset having 181 column, we selected a few to keep in our analysis process in order to keep this report concise. We “deleted” all the columns except for #1, #2, #3, #6, #9, #10, #15, #16, #17, #21, #39, #40, and #41
   1. #1 Show ID – the unique id for each show.
   2. #2 Show Time – the time the show airs.
   3. #3 Show Conical Name – the full name of the show.
   4. #6 Show Episode Count – the number of episodes for each show.
   5. #9 Show Genre – the genre of each show.
   6. #10 Show Sub Genre – each possible genre of each show.
   7. #15 Show Name – the name of each show as it appears on air.
   8. #16 Show Rating – the rating from 1-5 of each show.
   9. #17 Show Season Count- the number of seasons for each show.
   10. #21 Show Type – the type of show.
   11. #39 Show Company ID – the unique id of the company where the show is aired.
   12. #40 Show Company Channel ID- the unique id of the channel the show is aired on.
   13. #41 Show Company Name – the full name of the company that owns the show.
3. The original dataset had longer and more confusing column names, so our team renamed them to be simplified and easier to read.
4. Next, our team removed any weird symbols from the showType column and removed the “~” symbol from the showSubGenre column, as well as converted the showTime into a readable time of show.

**Descriptive Statistics and Visualization**

**Histogram 1**

Show season count: This histogram shows that as season numbers increase, the overall episodes per season tend to decrease.

Chart, histogram

Description automatically generated

**Histogram 2**

Show Episode count: This histogram shows the general count of episodes. We can see that most shows remain in the lower episode count range.

Chart, histogram

Description automatically generated

**Histogram 3**

Show rating: Since our overall goal is to assess show rating, we created a histogram of show rating. Since the dataset we originally pulled from is the 1000 most popular shows on Hulu, we can see that show ratings remain at about a 3 or higher.

Chart, bar chart

Description automatically generated

**Plot 1**

Season Episode Count: This histogram plots show episode count based on show season count and show genre. For each genre, we can see what shows had the most seasons and most episodes. We can see a trend where, as the number of seasons increases, the number of episodes increases as well.

Chart, scatter chart

Description automatically generated

**Plot 2**

Season episode count 2: Since most shows that are popular have a lot of episodes, we also decided to condense our view and plot shows that had 250 episodes or more. With this condensed view, we can see that there is less of a pattern once we look at shows with more than 250 episodes.

**Chart, scatter chart

Description automatically generated**

**Plot 3**

Show Genre vs Show Rating: This plot shows us what shows within their respective genres received the highest show ratings. From this plot we can see that “Drama” and “Comedy” genres contained the most shows with the highest ratings.

**Scatter chart

Description automatically generated with low confidence**

**Plot 4**

Show Episode Count vs Show Rating: This plot shows us what episode counts have the highest show ratings. From this plot we can see that as episode count increases, show ratings tend to also be higher.

**Chart, scatter chart

Description automatically generated**

**Plot 5**

Show Season Count vs Show Rating: This plot shows us what shows have the highest show ratings based on season count. From this plot we can see that as season count increases, show ratings tend to also be higher.

Chart, scatter chart

Description automatically generated

**Plot 6**

Show Company vs Show Rating: This plot shows us what shows have the highest show ratings based on the company they belong to. From this plot we can see a that companies such as Nickelodeon, FOX and ABC, have more shows with higher ratings.

**Chart

Description automatically generated with medium confidence**

**Data Modeling**

**Linear Regression Model**

**Introduction:**

In total we ran seven linear regression models to rest the relationships between independent and dependent variables. Based on these models, we discovered what independent variables affected our dependent variable of show rating.

**Model 1**

In our first model, we put show rating as y, and show episode count as our x variable. We got an R2 of 0.0004533 and adjusted R2 of -0.0005482, showing us that there is not a strong, or significant relationship between show rating and show episode count.

Text, letter

Description automatically generated

**Model 2**

Our second model looks at show ratings as our y variable and show genre as our x variable. We got an R2 of 0.4843 and adjusted R2 of 0.4665, which, compared to our model 1, is statistically significant. We can take a closer look at our intercepts and see that certain columns are also more statistically significant than other columns. From that perspective, showGenreClassics and showGenreScience Fiction are less statistically significant than other aspects within show genre.

**Table

Description automatically generated**

**Model 3**

Model 3 looks at show rating versus show season count, being our y and x variables respectively. We get an R2 of 4.283e-07 and adjusted R2 of -0.001002, which, similar to model 1, is not statistically significant when we compare our x variable to our y variable.

**Text, letter

Description automatically generated**

**Model 4**

Our fourth model looks at the show rating as our y variable and show company id as our x variable. This model produces an R2 of 0.00948 and adjusted R2 of 0.008488, which is not statistically significant, showing us that show company id does not affect the outcome of show ratings as much as other models.

Text, letter

Description automatically generated

**Model 5**

This model looks at show rating versus show company channel id, as our y and x variables. The R2 and adjusted R2 values of 0.01081 and 0.009818 are more statistically significant than other models. This shows, if we company model 5 to model 4, that show channel id has more of an effect on the outcome of show ratings than show company id does.

**Text, letter

Description automatically generated**

**Model 6**

Model six looks at show ratings as the y variable and show company name as our x variable. We got an R2 of 0.3755 and adjusted R2 of 0.3541, which is statistically significant. We can also look at the intercepts of the columns belonging to show company id and see that showCompanyNameMTV and showCompanyNameABC are among some of our statistically significant columns compared to showCompanyNameDisney Junior which is not statistically significant.

**Table

Description automatically generated**

**Graphical user interface, text, application, table

Description automatically generated**

**Table

Description automatically generated**

**Model 7**

Our final model is our multiple regression model looking at show rating, as the y variable, versus show company id, show company channel id and show company name as our x variables. This multiple regression model produces an R2 of 0.3755 and adjusted R2 of 0.3541. This happens to be the same values from our model 6, which gives an idea that show company name has more of an effect on show rating compared to the other variables. Looking at model 7 as a whole, it is still statistically significant, meaning our x variables together result in a large effect on the outcome of show rating. We can also look at the combined aspects of the variables to see that showCompanyNameHulu Original Series and showCompanyNameAdult Swim are among some of the more statistically significant columns within our variables.

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

**IST 707 HW #3**

**Introduction**

In 1989, UK banks introduced the Personal Equity Plan (PEP), targeting individuals over 18 to invest primarily in British companies. PEPs provided tax-free accumulation of income and capital gains from shares, unit trusts, or investment trusts, making them highly appealing. This sparked a surge in potential clients seeking tax-free investment income.

However, alongside PEPs' introduction came the challenge of identifying eligible clients for this investment opportunity. This report analyzes bank data containing 11 initial client attributes, including demographic and banking information. These attributes will be used to assess clients' eligibility for the new Personal Equity Plan. Through Association Rule discovery on the bank data, the analysis seeks to identify strong rules indicating a customer's suitability for a PEP.

**Analysis and Models**

The first stage involves integrating and configuring the necessary libraries and packages into the script file. Following this, the data will be imported from a CSV file and organized into a data frame named "bankdata." Utilizing the "str()" function, an examination of the data was conducted, leading to the following observations:

A screenshot of a computer code

Description automatically generated

**Data Cleaning**

Before proceeding to the subsequent data processing step, it is imperative to eliminate, discretize, or convert certain unnecessary data. Initially, the "id" attribute and all instances of missing values (NAs) within the dataset were removed.

A close-up of a computer code

Description automatically generated

A computer code with black text

Description automatically generated

Two variables, income and age, underwent discretization. Income was categorized into three groups: "LowIncome," "MidIncome," and "HighIncome." Age was divided into seven groups: "Child," "Teen," "Twenties," "Thirties," "Forties," "Fifties," and "Senior." Additionally, the "children" attribute was transformed from numeric to factor format.

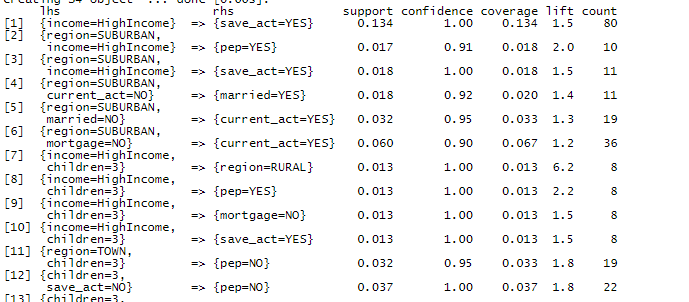
A math equation on a white background

Description automatically generated

A close-up of a word

Description automatically generated

Several Association Rule Discovery tests were conducted (totaling 7 tests), yielding the following outcomes:



A diagram of a network

Description automatically generated

**Insights**

1. Clients who have either a savings account or a checking account, with one child, and are in their “Forties” have a very high chance of qualifying for a PEP.
2. Teenagers are 70% NOT likely to be considered for a PEP (Must also be at least 18 years of age).
3. People with a single child have an 81% chance of being approved for a PEP.
4. Individuals who earn a high income and live in a suburban area have a high chance of being approved for a PEP.
5. Clients with a high income and more than one child have over 96% probability of favorable approval for a PEP.

* Python
  + Display use of Python to transform structured data by:
* Extracting data from a structured data source
* Process the data for analysis
* Creating graphs
* Recognize patterns and attributes that contribute to car value depreciation and how to predict future resale value
  + Display my use of Python to optimize a stock portfolio data by:
* Web scraping stock data
* Create reports of the stock data and create the optimal weighted average percent of stocks that should be in a stock portfolio to generate highest returns

**IST 652 Final Project**

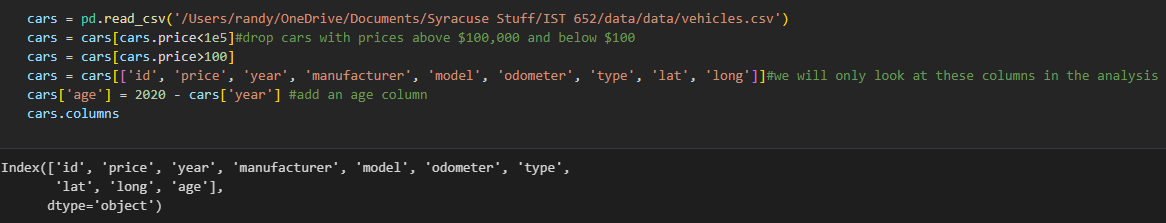
**Introduction**

This analysis is aimed at discovering which top 25 vehicles have the best resale value after a minimum of 2.5 years of ownership.

Install seaborn for statistical visualization and import necessary modules for data analysis and more visualizations.

A screenshot of a computer

Description automatically generated



The previous follows a sequential structure, where each line of code performs a specific operation on the dataset. Let's break down the code step by step:

Reading the Dataset: The first line of code reads the car data from a CSV file using the read\_csv() function from the Pandas library. The file path is specified as '/Users/randy/OneDrive/Documents/Syracuse Stuff/IST 652/data/data/vehicles.csv'.

Filtering by Price: The next two lines of code filter out cars with prices above $100,000 and below $100. This is achieved by using conditional filtering with the price column of the dataset.

Column Selection: The following line of code selects specific columns for analysis. The columns included are 'id', 'price', 'year', 'manufacturer', 'model', 'odometer', 'type', 'lat', and 'long'. By selecting only these columns, we can focus on the relevant information for our analysis.

Adding an Age Column: The last line of code adds an 'age' column to the dataset. This is done by subtracting the 'year' column from the current year (2020). The resulting 'age' column represents the age of each car in years.

**Data Cleaning**

A screenshot of a computer program

Description automatically generated

The provided code defines a function, get\_missing\_info, designed to analyze and report missing information within a DataFrame. The function conducts a comprehensive assessment by calculating various metrics, including the total number of entries, the number of null entries, the percentage of empty cells, the number of missing values in each column, the percentage of missing values in each column, the mode of each column, and the percentage of values matching the mode.

Upon execution, the function prints the total empty percentage and identifies columns where more than 97% of the values match the mode. The calculated information is then compiled into a new DataFrame, missing\_value\_df, which is subsequently returned by the function.

This function serves as a valuable tool for data analysts and researchers, facilitating a nuanced understanding of missing data patterns and aiding in decision-making processes related to data cleaning and imputation strategies.

A black rectangular object with white border

Description automatically generated

A computer screen shot of a program code

Description automatically generated

The code is made up of three functions and a few lines of code to apply these functions to the dataset.

The modef(x): function takes a group of values as input (x) and returns the mode of that group. It uses the pd.Series.mode() function to calculate the mode. If there is only one mode, it returns that mode. If there are no modes (i.e., all values are unique), it returns 'unknown'. If there are multiple modes, it returns the first mode.

The isna(x): function checks if a value (x) is NaN (not a number) or not. It uses the math.isna() function to check if the value is NaN. If the value is NaN, it returns True. Otherwise, it returns False.

The fill\_type(x): function fills the missing values in the "type" column based on the corresponding "model" group. If the "type" value is NaN, it tries to find the mode of the "model" group using the model\_types dictionary. If the mode is found, it returns the mode. If the mode is not found or the "type" value is not NaN, it returns the original "type" value.

The code aims to fill missing values in the "type" column of a dataset using the mode of the corresponding "model" group.

**Data Visualization**

A screen shot of a computer code

Description automatically generated

A graph of a number of vehicles

Description automatically generated

#plot pricing probability density for different types of vehicle

cars\_plt = cars[cars.type.isin(['sedan', 'SUV', 'truck'])]

sns.displot(cars\_plt, x='price', hue='type', kind='kde', bw\_adjust=0.6, cut=0, common\_norm=False, height=5, aspect=2)

plt.xticks(range(0,int(1e5), int(1e4)), labels=['${:,}'.format(x) for x in range(0, int(1e5), int(1e4))])

plt.xlabel('Price ($)')

plt.xlim(0,int(1e5))

plt.ylabel('Normalized Probability Density')

plt.title('Price Density Distribution by Type (Under $100,000)')

plt.show()

A graph of a number of people

Description automatically generated with medium confidence

Analyzing the price distribution of vehicles is essential for understanding the affordability and popularity of different models.

The code explored the price density distribution for different types of vehicles using a density plot. By visualizing the probability density of vehicle prices, we can gain insights into the distribution patterns and compare them across different vehicle types. This analysis can be useful for understanding market trends, pricing strategies, and customer preferences in the automotive industry.

The Price Density diagram illustrates the smoothed distribution of prices for sedans, SUVs, and trucks, with each curve normalized to have an area under 1. Notably, sedans exhibit a pronounced skewness toward the lower price spectrum. In comparison, the SUV curve bears closer semblance to the sedan curve than the truck curve, suggesting a discernible trend wherein SUVs are increasingly adopted as the primary mode of transportation, colloquially referred to as the 'daily driver'.

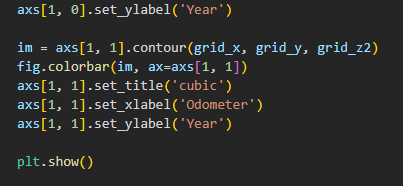
Trucks, in contrast, demonstrate a bifurcated pricing structure, featuring lower-end options priced below $10,000 alongside a substantial presence in the $30,000-$40,000 range. Across all vehicle categories, a discernible pricing strategy is observed, wherein prices often strategically hover just below multiples of $10,000. This pricing tactic is evidently employed to create a psychological perception of reduced cost, thereby potentially influencing consumer perceptions regarding the affordability of the respective vehicles.

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated



A screenshot of a graph

Description automatically generated

The production of a contour plot juxtaposing price against odometer reading and year provides a comprehensive overview of vehicular depreciation trends throughout their lifespan. However, the complexity of generating such a plot is compounded by the inherent noise and sparsity characterizing the pricing dataset. Specifically, for a given combination of year and odometer reading, the dataset exhibits multiple discrete price points, and the data distribution lacks a structured grid pattern. To address this challenge, an initial step involved the interpolation of pricing data, subsequently augmented by a smoothing process employing a moving average filter.

A black screen with white text

Description automatically generated

A screen shot of a computer program

Description automatically generated

A graph of average pricing data

Description automatically generated

Benchmark Depreciation rate: $0.18/mi

According to the data analysis, a newly acquired vehicle undergoes an approximate depreciation of 45% (equivalent to $16,000) over a 20-year period in the absence of any driving activity. In contrast, a comparable vehicle, subjected to an annual mileage of 50,000 miles, experiences a depreciation of 30% ($10,000) within a single year. To provide a practical context, the plotted arrow delineates the depreciation trajectory for a vehicle driven at an annual rate of 13,500 miles, reflective of the average mileage in the United States, over a span of 10 years. This scenario results in a total depreciation of 72% ($34,000 to $11,000), corresponding to a depreciation rate of $0.18 per mile.

A computer screen with text on it

Description automatically generated

A graph of a number of vehicles

Description automatically generated

#plot the average prices of the 25 most popular cars

com\_price = cars.loc[cars.make\_model.isin(com\_cars.index)]

ordered\_labels = com\_price.groupby('make\_model').price.median().sort\_values(ascending=False).index.values

fig, ax = plot.subplots(figsize=(12,6))

sns.boxenplot(data=com\_price, x="make\_model", y="price", order=ordered\_labels, ax=ax)

plt.xticks(rotation = 80)

plt.xlabel('Make and Model')

plt.ylabel('Price ($)')

plt.title('Pricing of the 25 Most Popular Vehichles')

plt.show()

A graph of blue and black objects

Description automatically generated with medium confidence

#plot the average prices of the 25 most popular trucks, trucks, and SUVS

for thing in ['sedan', 'truck', 'SUV']:

    com = cars[cars['type']==thing].make\_model.value\_counts()[0:25].index

    com\_price = cars.loc[cars.make\_model.isin(com)]

    ordered\_labels = com\_price.groupby('make\_model').price.median().sort\_values(ascending=False).index.values

    fig, ax = plot.subplots(figsize=(12,6))

    seab.boxenplot(data=com\_price, x="make\_model", y="price", order=ordered\_labels, ax=ax)

    plot.xticks(rotation = 80)

    plot.xlabel('Make and Model')

    plot.ylabel('Price ($)')

    plot.title('Pricing of the 25 Most Popular {}s'.format(thing.capitalize()))

    plot.show()

A graph with blue and black objects

Description automatically generated with medium confidence

A graph of blue and black lines

Description automatically generated with medium confidence

A graph of blue columns with black text

Description automatically generated

from scipy.optimize import curve\_fit

#fit an exponential to the toyota corolla data to determine how well it holds its value

def func(x, a, b):

    return a \* np.exp(-b\*x)#exponential function we will use to fit

def plot\_depr(data, func, model):

    #get model data and filter out cars older than 50 years

    df = data[(data['make\_model']==model) & (data['age']<=50)].sort\_values(by='age')

    xdata = df['age']

    ydata = df['price']

    #fit to the data

    popt, \_ = curve\_fit(func, xdata, ydata, p0=[4e4, 0.1])#fit the exponential to the data

    init = popt[0]#intiial value (age=0) according to the curve fit

    depr20 = -np.log(0.80)/popt[1]#time to depreciate 20% according to the curve fit

    depr90 = -np.log(0.10)/popt[1]#time to depreciate 90% according to the curve fit

    fig, ax = plot.subplots(figsize=(10,5))

    carplt = ax.scatter(xdata, ydata, c=df['odometer'], cmap='viridis')#scatter plot of age vs price, colored by odometer

    plot.plot(xdata, func(xdata, \*popt), 'r--')#plot the fitted curve

    plot.text(0.5, 0.85,

             'Initial Value: {:,.0f}$\n'

             'Time to lose 20% value: {:.2f} years\n'

             'Time to lose 90% value: {:.2f} years'.format(init, depr20, depr90),

             transform = ax.transAxes,

            bbox=dict(facecolor='white', edgecolor='black'))

    cbar=plot.colorbar(carplt)

    cbar.set\_label('Odomater (mi)')

    plot.xlabel('Age (years)')

    plot.ylabel('Price ($)')

    plot.title(model)

    plot.show()

A graph of a graph showing the value of a number of years

Description automatically generated with medium confidence

When purchasing a vehicle, considerations extend beyond the immediate cost as well as future resale values. In this context, a methodological approach involves fitting a decaying exponential function to each specific vehicle model, facilitating a quantitative assessment of its depreciation dynamics over time. This analytical framework enables the determination of whether a given model maintains its value over the course of ownership.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.optimize import curve\_fit

# Define the function for curve fitting

def func(x, a, b):

    return a \* np.exp(-b \* x)

# Run curve fits for the 25 most popular sedans, SUVs, and trucks and plot

for kind in ['sedan', 'SUV', 'truck']:

    com = cars[cars['type'] == kind].make\_model.value\_counts().index[:25]

    depr\_df = pd.DataFrame(columns=['Model', 'val0', 'depr20', 'depr90'])

    for name in com:

        df = cars[(cars['make\_model'] == name) & (cars['age'] <= 50)].sort\_values(by='age')

        xdata = df['age']

        ydata = df['price']

        popt, pcov = curve\_fit(func, xdata, ydata, p0=[4e4, 0.1])

        init = popt[0]

        depr20 = -np.log(0.80) / popt[1]

        depr90 = -np.log(0.10) / popt[1]

        # Use pd.DataFrame instead of DataFrame if there's a conflict with another variable named pd

        depr\_df = pd.concat([depr\_df, pd.DataFrame({'Model': [name], 'val0': [init], 'depr20': [depr20], 'depr90': [depr90]})], ignore\_index=True)

    depr\_df = depr\_df.sort\_values(by='depr20', ascending=False)

    fig, ax = plt.subplots(figsize=(12, 5))

    sns.barplot(data=depr\_df, x='Model', y='depr20', ax=ax)

    plt.title('Depreciation of the 25 Most Popular {}s'.format(kind.capitalize()))

    plt.ylabel('Time to Depreciate 20% (Years)')

    plt.xticks(rotation=80)

    plt.show()

A graph of blue bars with white text

Description automatically generated

A graph of a number of people

Description automatically generated with medium confidence

A graph of a number of people

Description automatically generated with medium confidence

# Define the function for curve fitting

def func(x, a, b):

    return a \* np.exp(-b \* x)

# Boxplot of depreciation for different manufacturers

makes = cars['manufacturer'].value\_counts()[:25].index  # 15 most popular manufacturers

depr\_df = pd.DataFrame(columns=['Make', 'Model', 'val0', 'depr20', 'depr90'])  # this will hold the depreciation data

for make in makes:

    com = cars[cars['manufacturer'] == make].model.value\_counts()[0:10].index  # get the 10 most popular models by the manufacturer

    for name in com:

        df = cars[(cars['model'] == name) & (cars['age'] < 50)].sort\_values(by='age')  # get data for the model for ages under 50

        xdata = df['age']

        ydata = df['price']

        popt, pcov = curve\_fit(func, xdata, ydata, p0=[4e4, 0.1])  # fit to the data

        init = popt[0]  # initial value

        depr20 = -np.log(0.80) / popt[1]  # time to depreciate 20%

        depr90 = -np.log(0.10) / popt[1]  # time to depreciate 90%

        # Use pd.DataFrame instead of DataFrame if there's a conflict with another variable named pd

        depr\_df = pd.concat([depr\_df, pd.DataFrame({'Make': [make], 'Model': [name], 'val0': [init], 'depr20': [depr20], 'depr90': [depr90]})], ignore\_index=True)

# order the data in terms of decreasing median depreciation time

order = depr\_df.groupby('Make')['depr20'].median().sort\_values(ascending=False).index

fig, ax = plt.subplots(figsize=(12, 5))

sns.boxenplot(data=depr\_df, x='Make', y='depr20', order=order)

plt.title('Depreciation of the 25 Most Popular Makes of Vehicle')

plt.ylabel('Time to Depreciate 20% (Years)')

plt.ylim(0.5, 4)

plt.xticks(rotation=90)

plt.show()

A graph of a graph of a number of people

Description automatically generated with medium confidence

In the United States, empirical data indicates that Jeeps and Rams exhibit a comparatively robust retention of value, suggesting a sustained desirability for larger vehicles among consumers. Notably, certain older Jeep models have garnered heightened demand, leading to instances where their market value may appreciate over time. Concurrently, Toyota vehicles also demonstrate notable resilience in retaining value, with a median depreciation period of approximately 2.2 years required for a vehicle to undergo a 20% depreciation.

A screen shot of a computer program

Description automatically generated

A map of the united states

Description automatically generated

#plot the distribution of sedans, trucks, and suvs in the area around Vancouver

import geoplot

extent = (-1.5e7,-0.7e7, 2.6e6,6.5e6) #x and y limits of where we will plot

#look at sales of trucks, sedans, and SUVS

gtrucks = gdf[(gdf['type']=='truck')]

gsedans = gdf[(gdf['type']=='sedan')]

gsuvs = gdf[(gdf['type']=='SUV')]

#get the longitude data for each type of vehicle

trucks= pd.Series(gtrucks.geometry.x)

sedans= pd.Series(gsedans.geometry.x)

suvs= pd.Series(gsuvs.geometry.x)

fig, ax = plot.subplots(2, figsize=(13, 8), gridspec\_kw={'height\_ratios': [1, 6], 'hspace':0})

#plot the density distribution of each type by longitde

sns.kdeplot(data=trucks, ax=ax[0], clip=(extent[0], extent[1]), bw\_adjust=0.35, label='Trucks', color='red')

sns.kdeplot(data=sedans, ax=ax[0], clip=(extent[0], extent[1]), bw\_adjust=0.35, label='Sedans', color='blue')

sns.kdeplot(data=suvs, ax=ax[0], clip=(extent[0], extent[1]), bw\_adjust=0.35, label='SUVs', color='green')

ax[0].set\_xlim(extent[0], extent[1])#set the x limits of the plot to be the same as our map

ax[0].set\_ylabel('Probability Density')

ax[0].set\_title('Sales of Trucks and Sedans in the Lower 48 States')

ax[0].set\_xticks([])

ax[0].legend()

#Plot the location of each posting on a map

geoplot.pointplot(gtrucks, ax=ax[1], s=1, color='red')

geoplot.pointplot(gsedans, ax=ax[1], s=1, color='blue')

geoplot.pointplot(gsuvs, ax=ax[1], s=1, color='green')

ax[1].axis(extent)

ctx.add\_basemap(ax[1], zoom=4)

plt.show()

A map of the united states

Description automatically generated

**FIN 554 HW #3 Portfolio Optimization**

We created a function to leverage QuantStats Library to analyze and visualize investment performance metrics. We are also using yfinance to download historical stock data for specified symbols and time periods. The function calculates daily returns from adjusted closing prices and then aggregates them into monthly returns. We also create a variable called PAST that looks at past monthly returns for a group of stocks at a specific timeframe.

A screen shot of a computer program

Description automatically generated

Riskfolio-Lib is a Python library that provides tools for portfolio optimization and risk management.



A screen shot of a computer screen

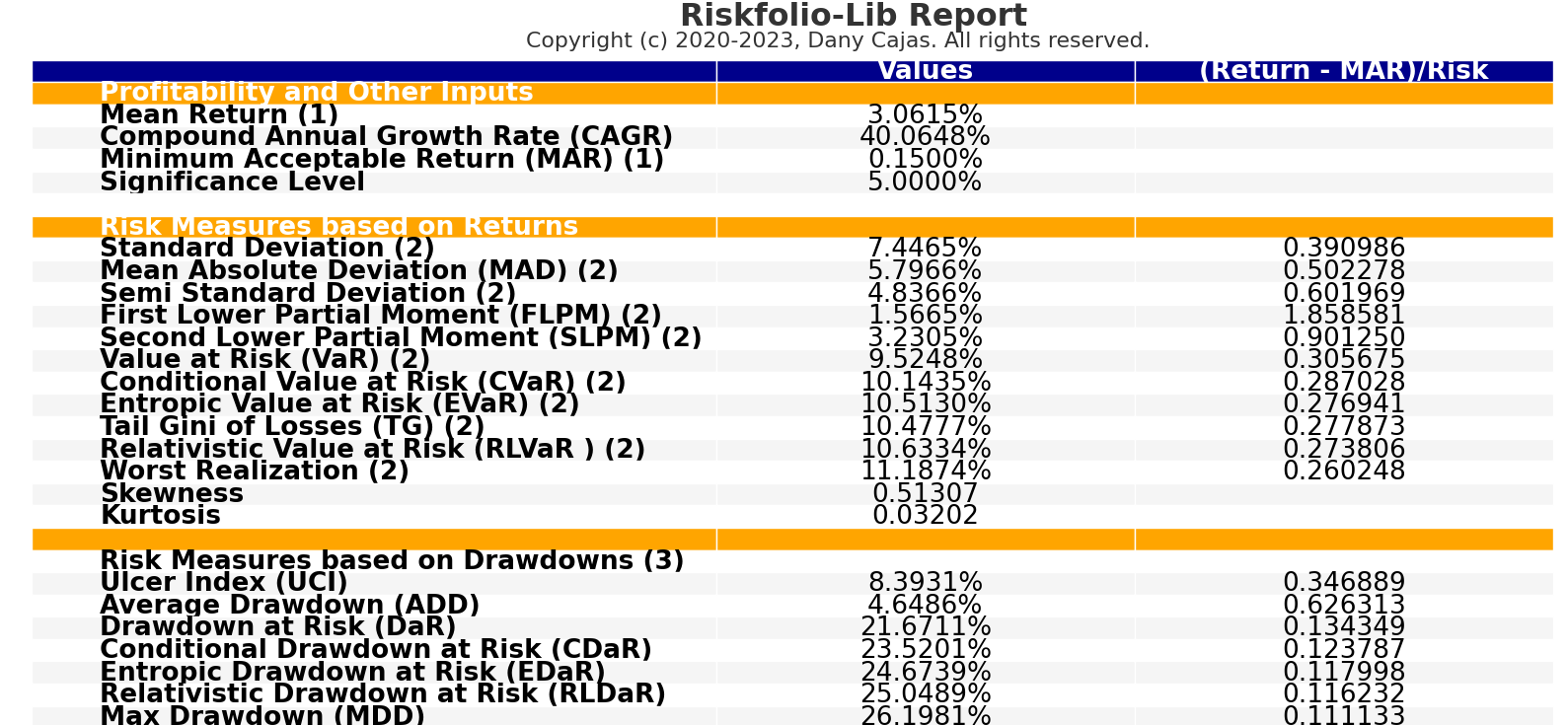
Description automatically generated

Python code demonstrates the process of portfolio optimization using the OWL library. By initializing a portfolio object, computing asset statistics, and optimizing the portfolio based on the Sharpe ratio, investors can make informed decisions to achieve optimal risk-adjusted returns. Understanding such code snippets is essential for financial analysts and investors looking to enhance their portfolio management strategies.

**Generated Visuals for Portfolio optimization:**

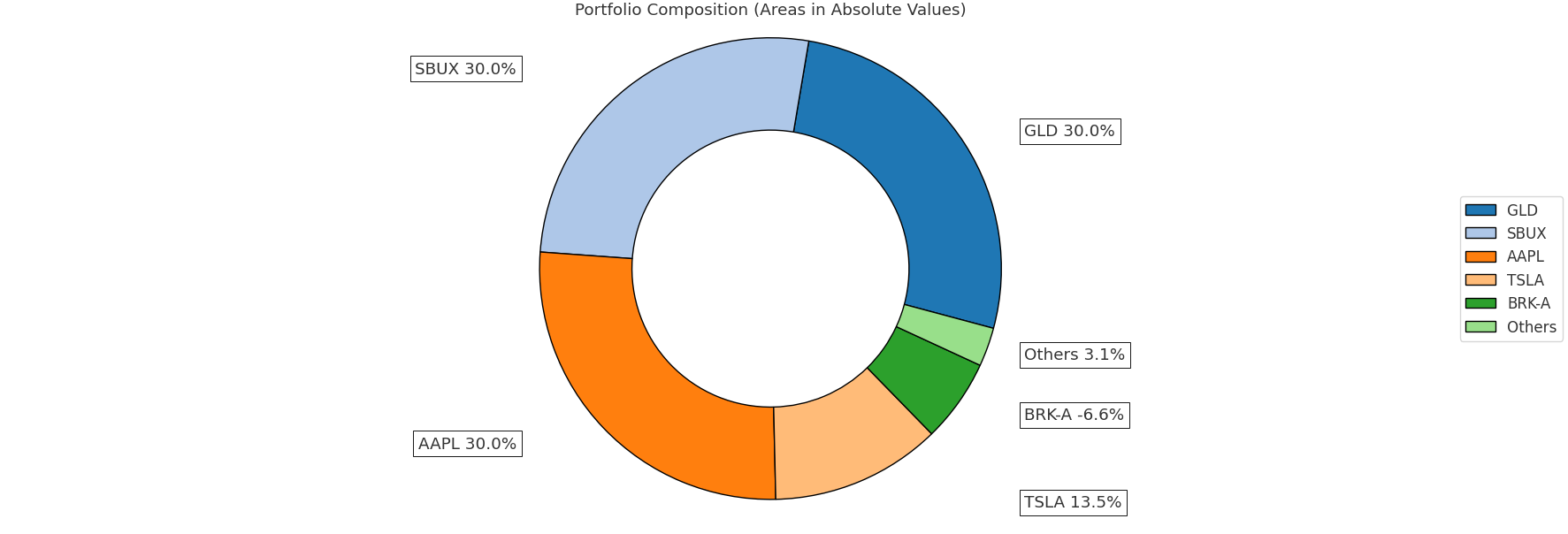
Python Code to Generate Reports and graphs.





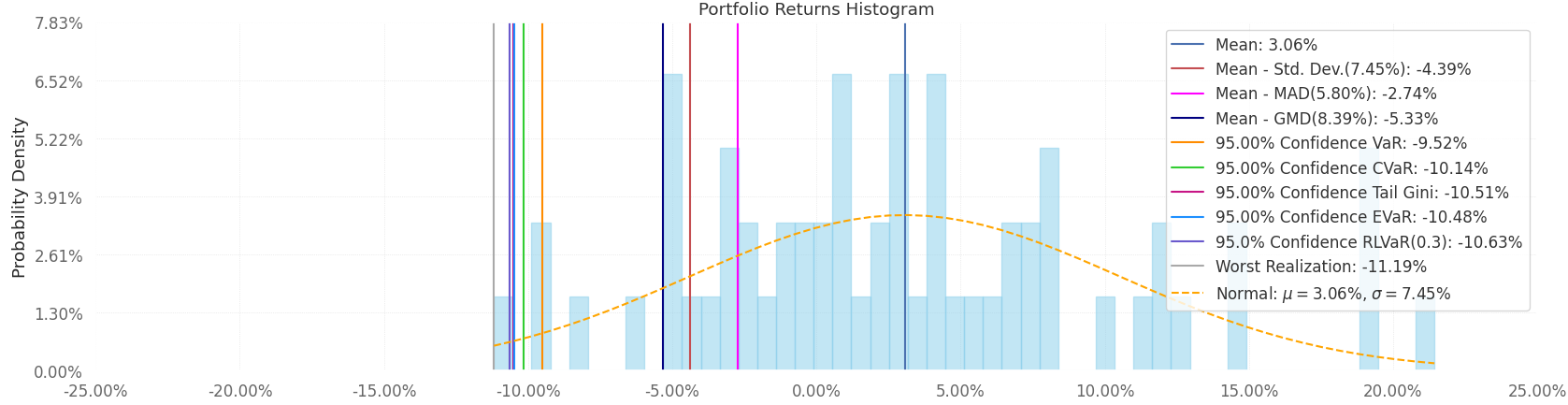
A graph showing a line

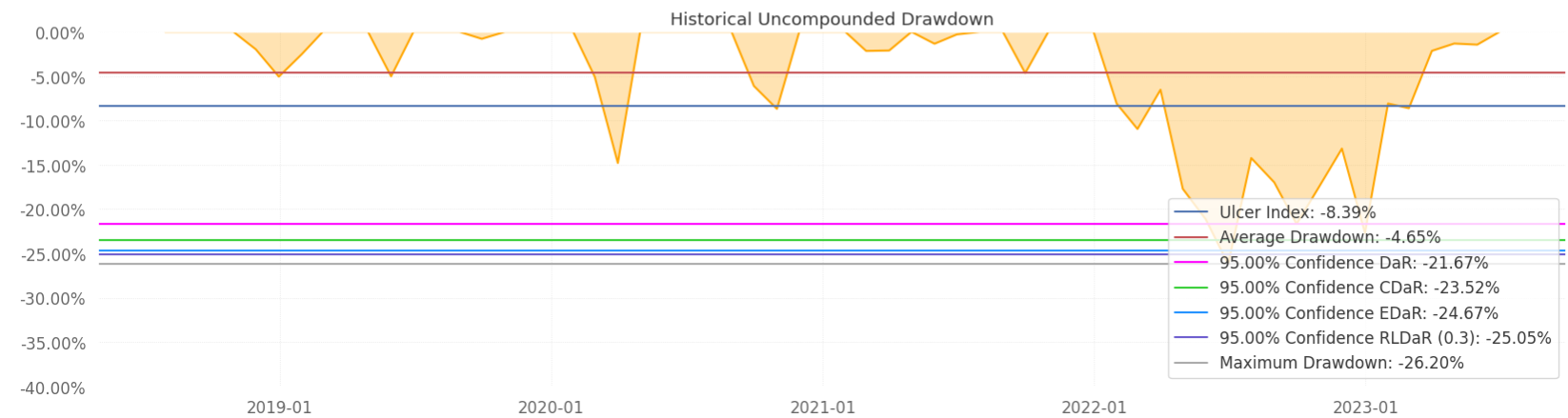
Description automatically generated



A screenshot of a graph

Description automatically generated





**Optimized Portfolio recommendation**

**A screen shot of a computer

Description automatically generated**

Python code showcases the process of portfolio optimization using the OWL library. By setting parameters, calculating statistics, and optimizing the portfolio, the code aims to find the optimal asset mix based on the mean-variance approach. Understanding and implementing such optimization techniques are essential for effective portfolio management in the financial domain. Below is a table showing the recommended stock weights to optimize a portfolio and maximize returns based on the given stocks. Our recommendation is as follows:

1. The following stocks should each make up 30% of the portfolio: Apple (AAAPL), Gilead (GLD), and Starbucks (SBUX).
2. Remove Berkshire Hathaway Class A Stock, 6.93% of the portfolio should Tesla (TSLA) stock.
3. The remaining 3.07% of the portfolio should consist of Dogecoin (DOGE-USD).

A screenshot of a graph

Description automatically generated