

A
PROJECT DOCUMENTATION
ON
Benford Data Analysis



SUBMITTED BY
Anish Shilpakar

SUBMITTED TO
Coderush Nepal

April 23, 2023

TABLE OF CONTENTS

INTRODUCTION	3
TOOLS AND TECHNOLOGIES USED	5
METHODOLOGY	6
CONCLUSION	16

INTRODUCTION

This is the challenge project 2 in the Coderush Data Engineering Apprenticeship program. Here in this project I have used various tools and technologies to complete the given project tasks as mentioned in the objectives section. This is a simple web app project developed using Pyramid framework of Python. Here the frontend of web application presents user with a input form, where user can upload a dataset as a csv file. Then it is checked if the dataset satisfies Benford's law or not. If the uploaded dataset satisfies the Benford's law JSON output is returned to the user.

Benford's Law

Benford's Law, also known as the First-Digit Law, is an observation about the frequency distribution of digits in many naturally occurring datasets. The law states that in many datasets, the first digit is more likely to be small, with the frequency of digits decreasing as the value of the digit increases. Specifically, the probability of a digit being the first digit of a number is approximately equal to the logarithm of one plus the reciprocal of that digit. Benford's law, also known as the Newcomb–Benford law, the law of anomalous numbers, or the first-digit law, is an observation that in many real-life sets of numerical data, the leading digit is likely to be small.

A set of numbers is said to satisfy Benford's law if the leading digit d ($d \in \{1, \dots, 9\}$) occurs with probability

$$P(d) = \log_{10} \left(1 + \frac{1}{d} \right)$$

The leading digits in such a set will have the following distribution known as Benford's distribution

d	1	2	3	4	5	6	7	8	9
P(d)	30.1%	17.6%	12.5%	9.7%	7.9%	6.7%	5.8%	5.1%	4.6%

Also, Benford's distribution can be visualized graphically using following figure

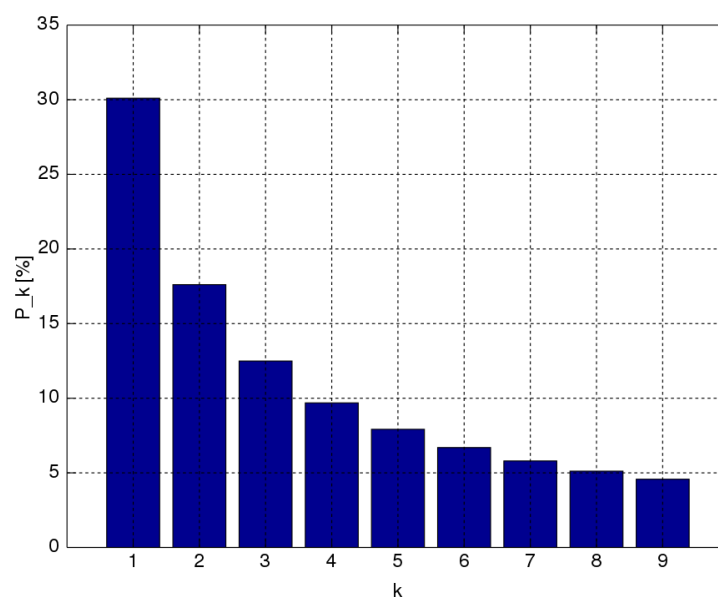


Figure 1: Distribution of first digits according to Benford's law

It has been shown that this result applies to a wide variety of data sets, including electricity bills, street addresses, stock prices, house prices, population numbers, death rates, lengths of rivers, and physical and mathematical constants. Also sequences like Fibonacci series, factorials, powers of 2 etc. follow the Benford's law. But there are some datasets like height data, students' grades, sequence of square roots, reciprocals that don't follow Benford's law. Benford's law is mostly used in accounting fraud detection, criminal trials, election data, macroeconomic data, price digit analysis, genome data etc.

In this project, we will compare the data distribution of the input dataset with the theoretical Benford's data distribution to check whether the input dataset follows Benford's law or not. This is further discussed in methodology section

PROJECT OBJECTIVES

- To create a web application in Python using Pyramid framework which has one endpoint /benford that accepts a csv file with just one column with 10k+ rows of numbers as input
- To produce a JSON output if the input csv conforms the Benford's law on first digits

SCOPE AND APPLICATIONS

The project can have following scope and applications

1. **Fraud detection:** Benford's Law has been used as a tool for detecting potential fraud in financial data, as well as other types of data. By checking if a dataset follows Benford's Law, your web application could help identify instances where the data may have been manipulated or falsified.
2. **Quality control:** In some industries, such as manufacturing or scientific research, data accuracy is critical. By checking if a dataset follows Benford's Law, your web application could help identify potential errors or inconsistencies in the data, allowing for improved quality control.
3. **Educational tool:** Benford's Law is an interesting statistical phenomenon that can be used to teach students about the distribution of digits in large datasets. Your web application could be used as an educational tool to help students understand the concept of Benford's Law and how it can be applied in real-world situations.
4. **Business analysis:** Many companies rely on large datasets to make important business decisions. By checking if a dataset follows Benford's Law, your web application could help companies ensure the accuracy and integrity of their data, leading to more informed decision-making.

TOOLS AND TECHNOLOGIES USED

1. Python

Python is a high-level, interpreted, and general-purpose programming language. Python offers various key features like simplicity, readability, versatility, etc. making it suitable for a wide range of applications like web development, scientific computing, data engineering, data analysis, data processing, machine learning and more. Also, Python is also an object-oriented programming language, meaning that it is built around the concept of objects and classes. This makes it easy to create complex and modular programs, as well as to reuse code and make modifications to existing code.

2. Pyramid

Pyramid is an open-source web framework for Python. It is designed to make it easy to write web applications by providing a set of tools and libraries that handle common tasks, such as URL routing, templating, and authentication. Pyramid is built on top of the WSGI (Web Server Gateway Interface) specification, which allows it to work with a variety of web servers and platforms. Pyramid is a "minimalistic" framework, meaning that it provides only the core features needed to build web applications, and allows developers to choose which additional libraries and tools they want to use. This flexibility makes Pyramid a good choice for a wide range of applications, from small personal projects to large enterprise-level applications. As mentioned in the project objective, I have used Pyramid to develop the web application.

3. Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. It provides data structures for efficiently storing large datasets and tools for working with them. Pandas was used for generating the data distribution for each digit in the dataset

4. JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of the JavaScript programming language, but is language-agnostic and can be used with many programming languages.

5. Matplotlib

Matplotlib is a data visualisation library in Python. It provides a high-level interface for creating static, animated, and interactive visualisations in Python. Matplotlib allows users to create a wide range of visualisations, including line plots, bar plots, scatter plots, histograms, pie charts, box plots, error bars, etc. It is a powerful tool for data exploration and analysis, and is widely used in data science and scientific computing. Matplotlib can be easily integrated with other libraries in the PyData ecosystem, such as Pandas, to enable the creation of complex visualisations with just a few lines of code. Matplotlib was used to visualize the data distribution

METHODOLOGY

1. WORKING FLOW OF THE PROJECT

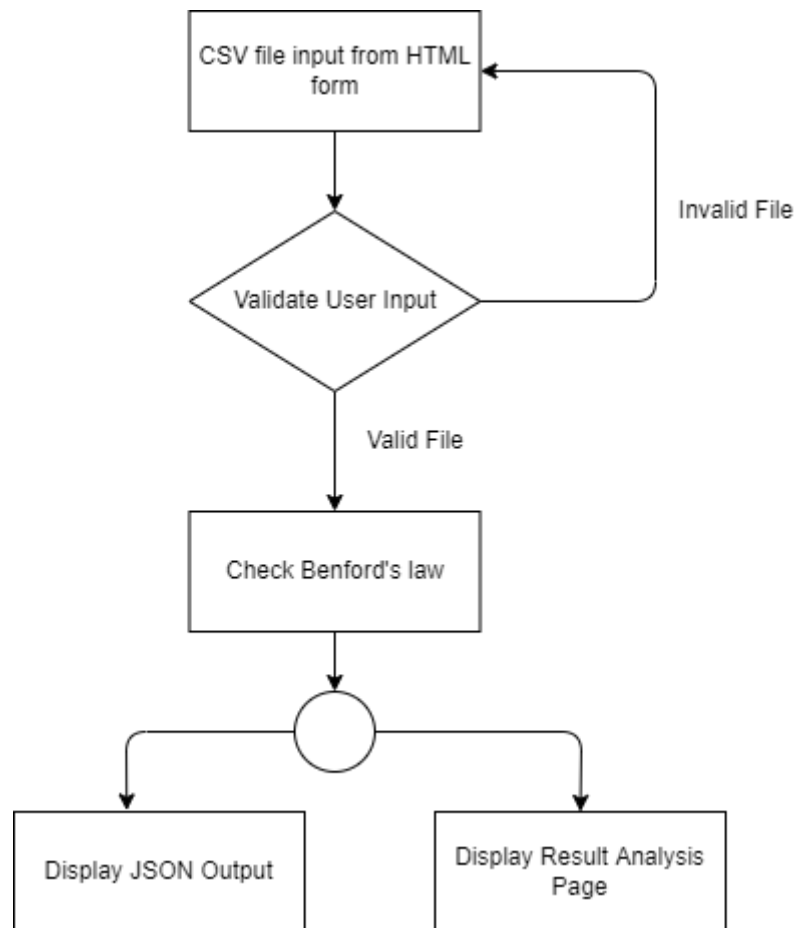


Figure 2: Flow chart showing the working flow of web app

The working flow of “Benford Data Analysis” includes following steps:

1. User input from HTML form
2. Input Validation
3. Check Benford’s law on Input dataset
4. Display JSON output
5. Display Result Analysis Page

1. User input from HTML form:

In this project the frontend of web app is developed using basic HTML, CSS and backend is developed using Pyramid framework. The homepage consists of an input form which is used to get the csv file as input from user. Here, user will upload a dataset in csv format which will then be checked for Benford's law. As the form fields have been set as required, so form can't be submitted until user has uploaded the file. Here user gets two options in the form. If user clicks "Get JSON" button, user will get JSON output for the input CSV data. If user clicks on Analyse button, user will be redirected to results page where user can compare the current data distribution with the Benford's data distribution. The input form is shown below:

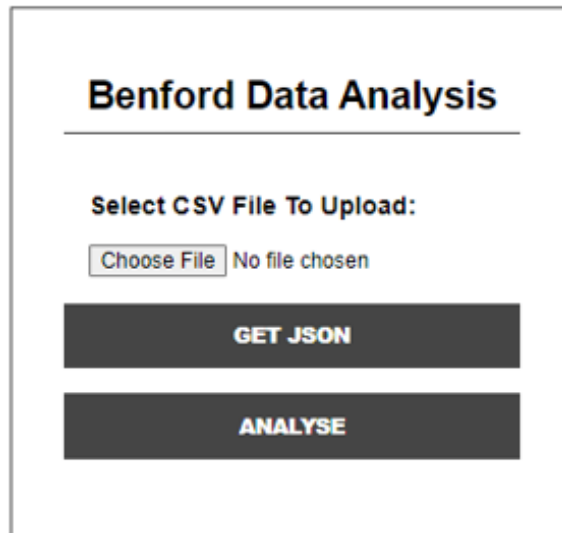
The image shows a web form titled "Benford Data Analysis". Below the title is a section labeled "Select CSV File To Upload:". This section contains a file selection interface with a button labeled "Choose File" and the text "No file chosen". Below the file selection area are two large, dark buttons: "GET JSON" and "ANALYSE".

Figure 3: User Input Form

2. Input Validation:

After user inputs file as input, the file is validated where we check if the file is actually a CSV file or not. If the input file is not a CSV file, error message is returned to the user.

```
# get the input from form
csv_file = request.POST['csvInput'].file
filename = request.POST['csvInput'].filename
# check if input files is CSV or not
name, ext = filename.split('.')
if not ext == 'csv':
    return {"Error": "Invalid File Input! Please upload CSV file"}
```

3. Check Benford's law on the input dataset:

In this step, we check if the given dataset follows Benford's law or not. For this, we firstly generate data distribution for given dataset (i-e dictionary with first digits as keys and their probabilities as values). When generating the data distribution, a bar plot is also created which help visualize the data distribution. The code to generate data distribution for a dataset is shown below:

```
# This function will return probability distribution of first significant
digit for the given dataset file
def get_data_distribution(filepath):
    # read csv
    df = pd.read_csv(filepath)
    # get the leading digit of all rows
    df['leading_digit'] = df.iloc[:,0].apply(lambda x:str(x)[0])
    # get value counts for each digit
    leading_digits_count = df['leading_digit'].value_counts()
    # convert to dictionary
    leading_digits_dict = leading_digits_count.to_dict()
    # filter out the non numeric values and 0
    filtered_dict = {}
    for k in leading_digits_dict:
        if isnumeric(k):
            if int(k) > 0:
                filtered_dict[k] = leading_digits_dict[k]
    # calculate probability distribution
    probability_dict = {k: filtered_dict[k]/sum(filtered_dict.values()) for
k in filtered_dict.keys()}
    # sort the dictionary for plotting
    sorted_dict = sort_dictionary_by_key(probability_dict)
    # plot bar graph for each digit and their frequency
    fig = plt.figure(figsize=(10,5))
    plt.bar(sorted_dict.keys(),sorted_dict.values())
    plt.xlabel("Leading Digits")
    plt.ylabel("Probability")
    plt.title("Probability distribution of first significant digit")
    plt.savefig("static/result_img.png")
    # return the observed distribution
    return probability_dict
```

After generating the data distribution for the input dataset, the theoretical Benford data distribution is also generated using the Benford's formula. Then the two distributions are compared to see if the input dataset follows Benford's law or not. The following is the code to generate Benford's data distribution.

```
benford_distribution = {str(k): np.log10(1+(1/k)) for k in range(1,10)}
```


To check for Benford's law, we firstly compute the absolute deviation between the probabilities (frequencies) of theoretical Benford distribution and the observed data distribution. We then check if the absolute deviation for all the digits is less than the limit or not. If the absolute deviation for all digits is less than the limit, the data follows Benford's law, else it doesn't follow Benford's law. For this limit was considered as 0.10. The code for checking if the distribution follows Benford's law is shown below:

```
def check_benford_law(filepath):
    benford_distribution = {str(k): np.log10(1+(1/k)) for k in range(1,10)}
    observed_distribution = get_data_distribution(filepath)
    # checking if the absolute deviation between numbers is within limit
    limit = 0.10
    ans = True
    differences = {}
    for k in benford_distribution.keys():
        differences[k] = abs(benford_distribution[k] -
observed_distribution[k])
        if differences[k] > limit:
            ans = False
    return ans
```

Also, I tried other methods to check if data distribution follows Benford's law or not. I compared the order of first digits in both distribution and the most frequent digit in both distributions. If the order was same in both distributions and the max frequent digit was found to be 1, then also, we can consider that the given data distribution follows Benford's law. Alternatively, Chi-square test can also be used to compare two distributions and determine if the given distribution follows Benford's law or not by setting a level of significance. For simplicity in this project, I have used the absolute deviation method to check for Benford's law.

4. Display JSON Output:

Our main task in this project was to create an endpoint called /benford that will check if the input csv conforms the Benford's law on first digits or not and then return a JSON output. This task was completed by creating a view called benford with the route "/benford" which takes the form input obtained from POST method. And then after input validation, we get the data distribution for the input dataset using get_data_distribution() function and then compare the data distribution with theoretical Benford's data distribution. Then based on result of comparison, we return a JSON output to the user. Also JSON file is saved locally. Here to return JSON output, we have used renderer as "json" in the pyramid view. The code for view named benford is shown below :

```
# to render the json result as output
@view_config(route_name="benford",renderer="json")
def benford(request):
    # get the input from form
    csv_file = request.POST['csvInput'].file
    filename = request.POST['csvInput'].filename
    # check if input files is CSV or not
    name, ext = filename.split('.')
    if not ext == 'csv':
        return {"Error": "Invalid File Input! Please upload CSV file"}
    else:
        # if it is CSV
        # save in local directory uploads
        filepath = f"uploads/{filename}"
        with open(filepath,"wb") as f:
            f.write(csv_file.read())
        # get data distribution for the input file
        data_distribution = get_data_distribution(filepath)
        # check if the distribution follows benford's law or not
        if check_benford_law(filepath) == True:
            message = "Given data follows Benford's law"
        else:
            message = "Given data doesn't follow Benford's law"
        # create a json file with the result
        result = {
            "Message": message,
            "Distribution": data_distribution
        }

        json_data = json.dumps(result)

        # save json file locally
        with open(f"output/{name}.json","w") as f:
            f.write(json_data)

    return result
```

5. Display Result Analysis Page:

This task wasn't mentioned in the task description, but I added this extra feature to the website for enhance usability of web app. I have added another page that will display the results of Benford's data analysis when user clicks on analyse button in the input form. The result analysis page will firstly show the result whether the current input csv dataset follows the Benford's law or not. Then it will also show the observed probability, actual Benford probability and absolute deviation for each digit. Similarly, for better understanding and visualization of user, this webpage will also display the barplot showing the data distribution of input dataset and also barplot for actual Benford distribution. So, by comparing these plots, user can get better idea whether the input dataset follows Benford's distribution or not. The elements of result analysis page are shown below:

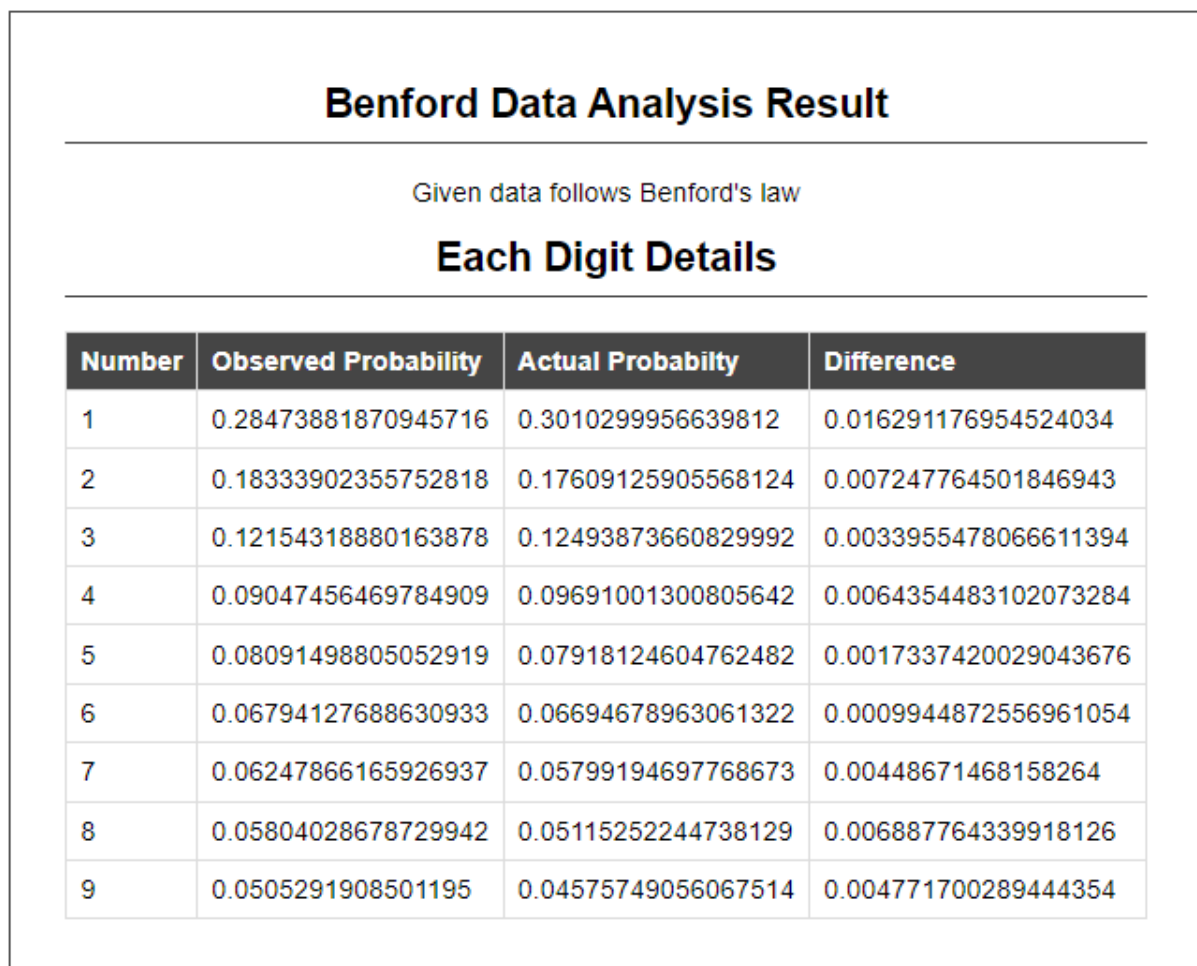
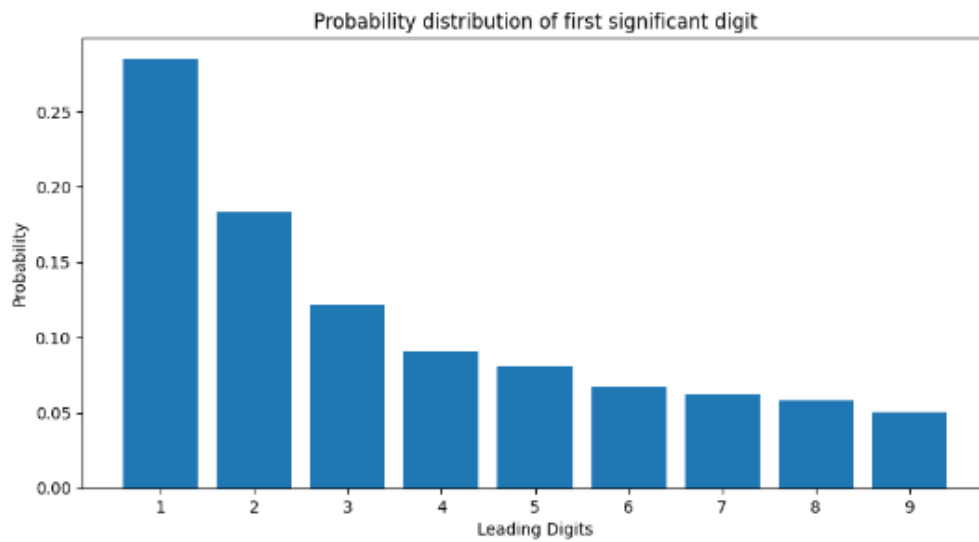


Figure 4: Comparison of the observed and actual Benford's data distribution

Observed Distribution For Given Data



Actual Benford Distribution

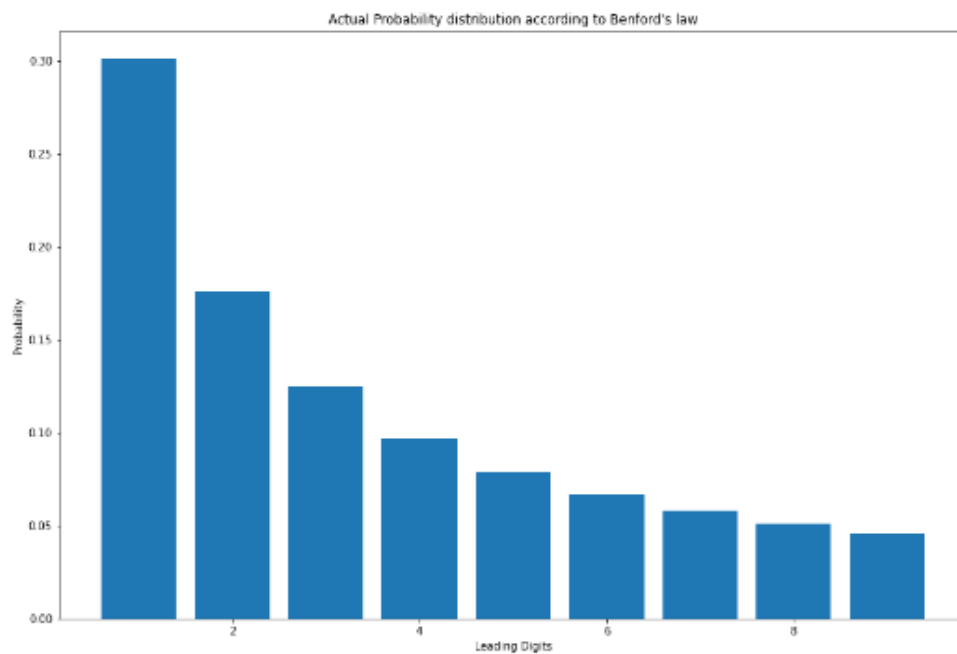


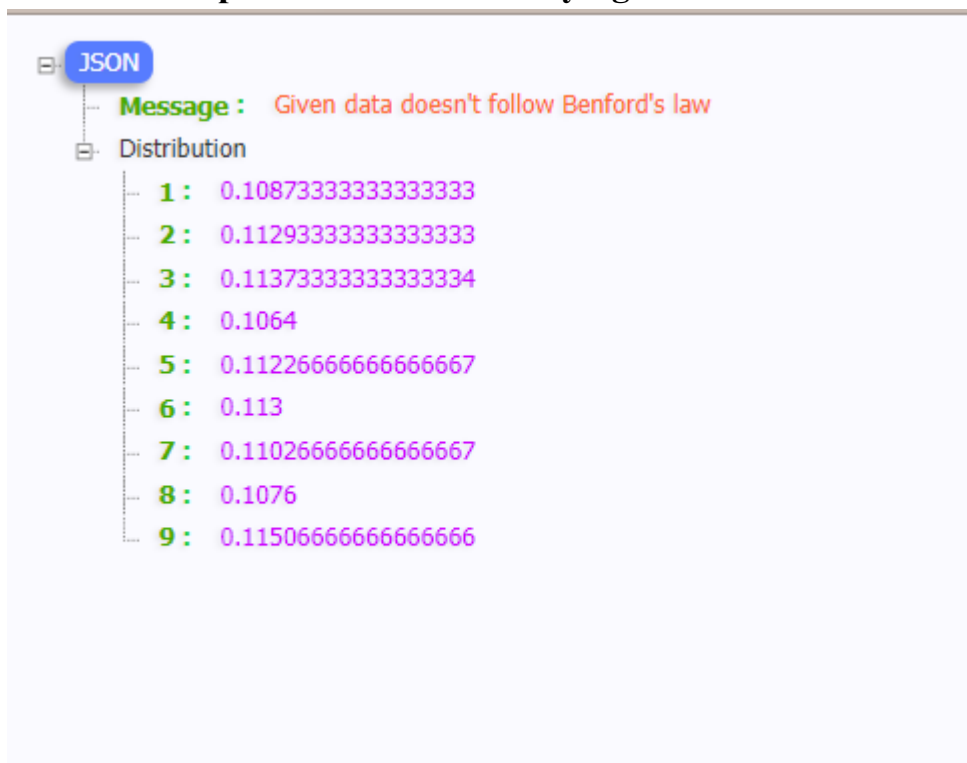
Figure 5: Visualization of observed and actual Benford data distribution

2. OUTPUT SCREENSHOTS

A. JSON Output for data satisfying Benford's law



B. JSON Output for data not satisfying Benford's law



C. Result Analysis Page for data satisfying Benford's law

[Back To Home](#)

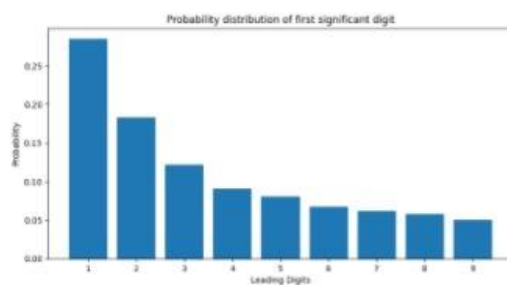
Benford Data Analysis Result

Given data follows Benford's law

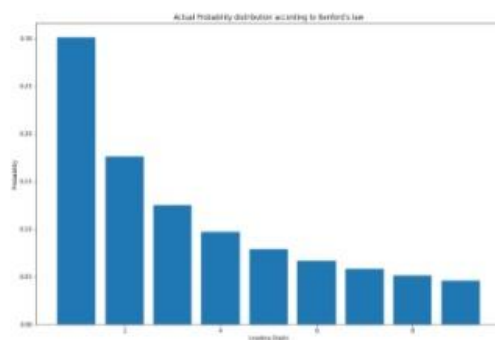
Each Digit Details

Number	Observed Probability	Actual Probability	Difference
1	0.28473881870945716	0.3010299956639812	0.016291176954524034
2	0.18333902355752818	0.17609125905568124	0.007247764501846943
3	0.12154318880163878	0.12493873660829992	0.0033955478066611394
4	0.09047456469784909	0.09691001300805642	0.0064354483102073284
5	0.08091498805052919	0.07918124604762482	0.0017337420029043676
6	0.06794127688630933	0.06694678963061322	0.0009944872556961054
7	0.06247866165926937	0.05799194697768673	0.00448671468158264
8	0.05804028678729942	0.05115252244738129	0.006887764339918126
9	0.0505291908501195	0.04575749056067514	0.004771700289444354

Observed Distribution For Given Data



Actual Benford Distribution



D. Result Analysis Page for data not satisfying Benford's law

[Back To Home](#)

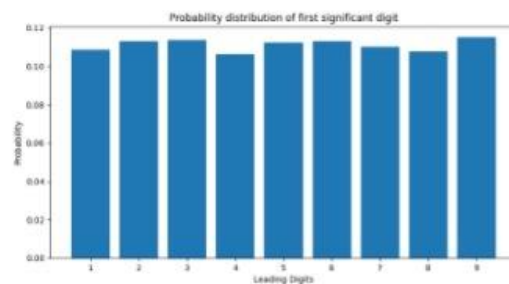
Benford Data Analysis Result

Given data doesn't follow Benford's law

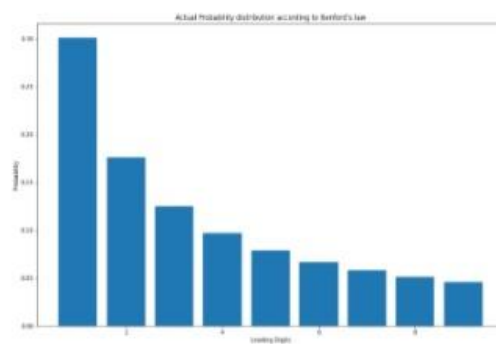
Each Digit Details

Number	Observed Probability	Actual Probability	Difference
9	0.11506666666666666	0.04575749056067514	0.06930917610599152
3	0.11373333333333334	0.12493873660829992	0.01120540327496658
6	0.113	0.06694678963061322	0.04605321036938678
2	0.11293333333333333	0.17609125905568124	0.06315792572234791
5	0.11226666666666667	0.07918124604762482	0.03308542061904185
7	0.11026666666666667	0.05799194697768673	0.05227471968897993
1	0.10873333333333333	0.3010299956639812	0.19229666233064785
8	0.1076	0.05115252244738129	0.05644747755261871
4	0.1064	0.09691001300805642	0.009489986991943575

Observed Distribution For Given Data



Actual Benford Distribution



CONCLUSION

To conclude, the challenge project for “Benford Data Analysis” was developed using Python and Pyramid framework and is working successfully. In this project, a Pyramid web application was developed, where users can input a csv dataset and then check if the dataset follows Benford’s law or not. To check if a dataset follows Benford’s law or not, firstly data distribution of input dataset is generated and then compared to the actual Benford’s data distribution using absolute deviation method. Here all the project objectives and tasks mentioned in the task description has been successfully completed. Additionally, I have added a result analysis page to increase understandability of the output generated by web app. The outputs for this project can be seen in outputs section. All necessary code for this project has been uploaded in GitHub.

The following are some future enhancements possible in this project.

1. Better UI and exception handling.
2. Using Benford’s analysis for Fraud detection

GitHub Project Link: [JuJu2181/Benford Data Analysis: Simple Project to check if dataset follows Benford's law or not using Python Pyramid framework \(github.com\)](https://github.com/JuJu2181/Benford_Data_Analysis: Simple Project to check if dataset follows Benford's law or not using Python Pyramid framework (github.com))