PROJECT REPORT

FRAUD DETECTION IN MEDICARE

SUBJECT: DAB422 CAPSTONE PROJECT II

Submitted by:

Davinder Kaur (W0813395)

Ronaldo Chan Fu Yi (W0831976)

Simerjeet Kaur (W0814800)

Table of Contents

Table of Figures

Table of Tables

1 INTRODUCTION

Health is a fundamental human right: "The enjoyment of the highest attainable standard of health is one of the fundamental rights of every human being without distinction of race, religion, political belief, economic or social condition" (WHO, 2017).

Advances in technology, medicine, new drugs and vaccines is propelling populations to have a higher life expectancy. United States doesn't have universal health coverage provided by the government, but for a few eligible beneficiaries, it is offered federal health insurance program: Medicare.

As a federal funded program, Medicare relies on tax contributions from the population to foment its service. The significant issue of fraud, waste, and abuse in the U.S. healthcare system estimate it to cost around $700 billion, hence fraud in healthcare systems increases the healthcare cost for the population, thus fraud detection and prevention are present-day challenges.

Our project is to propose a machine learning model to streamline the process to identify fraud and eligibility using publicized datasets made available by US federal agencies: Center for Medicare and Medicaid Services (CMS) and Office of Inspector General (OIG).

In our first attempt, we created a supervised machine learning model using features from Medicare dataset and target variable as eligibility from LEIE dataset. For this project, we intend to explore forensic accounting tool Benford's Law to yield additional insights for our dataset. We leveraged the dataset, exploring new interpretation through Benford's Law tests and creating further suspicions towards skewed data. We tried to address the following questions:

1. How real-world data behave when analyzed with forensic accounting tools Benford's Law?
2. In case we have discrepancies between real-world data and expected Benford's Law, how can we leverage this information to flag transactions as suspicious?
3. Can we create a deep learning feedforward neural network able to perform better than our benchmarked supervised learning model?

2 METHODS

For this project we will be working with a big volume dataset from Medicare part B dataset: 9,886,177 rows. Our project is limited by computing power, hence working with a huge load of data will be an additional challenge for our project.

Since this dataset is solely information on services and procedures provided to Original Medicare (fee-for-service) Part B (Medical Insurance) beneficiaries by physicians and other healthcare professionals, we will use additional datasets to create our target feature eligibility.

Integrating the additional datasets with Medicare dataset is expected to create few matches relative to the entirety of the data, hence working with an imbalanced dataset is a challenge, so in this case, we will try under sampling and over sampling techniques to enhance our model metrics.

We will take advantage of the work done previously and use one-hot encoded dataset, also use under sampling and over sampling when enhance our model metrics.

2.1 DATASETS

2.1.1 MEDICARE DATASET

For this project, we will be using the public available dataset: Medicare Physician & Other Practitioners - by Provider and Service with information on services and procedures provided to Original Medicare (fee-for-service) Part B (Medical Insurance) beneficiaries by physicians and other healthcare professionals; aggregated by provider and service.

It is important to notice that Medicare dataset has 29 columns with National Provider Identifier (NPI) and FFS data. NPI will be our key to match with the other datasets when creating target variable and FFS will give numbers to identify possible frauds or improper payments.

Dataset shape: 9.886.177 rows, 29 columns

Data dictionary for Medicare Part B:

Feature	Description
Rndrng_NPI	National Provider Identifier (NPI) for the rendering provider on the claim. The provider NPI is the numeric identifier registered in NPPES.
Rndrng_Prvdr_Last_Org_Name	When the provider is registered as an organization (entity type code = ‘O’), this is the organization name.
Rndrng_Prvdr_First_Name	When the provider is registered in NPPES as an individual (entity type code=’I’), this is the provider’s first name.
Rndrng_Prvdr_MI	When the provider is registered in NPPES as an individual (entity type code=’I’), this is the provider’s middle initial.
Rndrng_Prvdr_Crdntls	When the provider is registered in NPPES as an individual (entity type code=’I’), these are the provider’s credentials.
Rndrng_Prvdr_Gndr	When the provider is registered in NPPES as an individual (entity type code=’I’), this is the provider’s gender.
Rndrng_Prvdr_Ent_Cd	type of entity reported in NPPES. An entity code of ‘I’ identifies providers registered as individuals and an entity type code of ‘O’ identifies providers registered as organizations.
Rndrng_Prvdr_St1	The first line of the provider’s street address, as reported in NPPES.
Rndrng_Prvdr_St2	The second line of the provider’s street address, as reported in NPPES.

Rndrng_Prvdr_City	The city where the provider is located, as reported in NPPES.
Rndrng_Prvdr_State_Abrvtn	The state where the provider is located, as reported in NPPES. The fifty U.S. states and the District of Columbia are reported by the state postal abbreviation.
Rndrng_Prvdr_State_FIPS	FIPS code for rendering provider's state.
Rndrng_Prvdr_Zip5	The provider’s zip code, as reported in NPPES.
Rndrng_Prvdr_RUCA	Rural-Urban Commuting Area Codes (RUCAs), are a Census tract-based classification scheme that utilizes the standard Bureau of Census Urbanized Area and Urban Cluster definitions in combination with work commuting information to characterize all of the nation's Census tracts regarding their rural and urban status and relationships.
Rndrng_Prvdr_RUCA_Desc	Description of Rural-Urban Commuting Area (RUCA) Code
Rndrng_Prvdr_Cntry	The country where the provider is located, as reported in NPPES.
Rndrng_Prvdr_Type	Derived from the provider specialty code reported on the claim.
Rndrng_Prvdr_Mdcr_Prtcptg_Ind	Identifies whether the provider participates in Medicare and/or accepts assignment of Medicare allowed amounts.
HCPCS_Cd	HCPCS code used to identify the specific medical service furnished by the provider

HCPCS_Desc	Description of the HCPCS code for the specific medical service furnished by the provider
HCPCS_Drug_Ind	Identifies whether the HCPCS code for the specific service furnished by the provider is a HCPCS listed on the Medicare Part B Drug Average Sales Price (ASP) File.
Place_Of_Srvc	Identifies whether the place of service submitted on the claims is a facility (value of ‘F’) or non-facility (value of ‘O’).
Tot_Benes	Number of distinct Medicare beneficiaries receiving the service for each Rndrng_NPI, HCPCS_Cd, and Place_Of_Srvc.
Tot_Srvcs	Number of services provided; note that the metrics used to count the number provided can vary from service to service.
Tot_Bene_Day_Srvcs	Number of distinct Medicare beneficiary/per day services. Since a given beneficiary may receive multiple services of the same type (e.g., single vs. multiple cardiac stents) on a single day, this metric removes double-counting from the line service count to identify whether a unique service occurred.
Avg_Sbmtd_Chrg	Average of the charges that the provider submitted for the service.
Avg_Mdcr_Alowd_Amt	Average of the Medicare allowed amount for the service; this figure is the sum of the amount Medicare pays, the deductible

	and coinsurance amounts that the beneficiary is responsible for paying, and any amounts that a third party is responsible for paying.
Avg_Mdcr_Pymt_Amt	Average amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item service.
Avg_Mdcr_Stdzd_Amt	Average amount that Medicare paid after beneficiary deductible and coinsurance amounts have been deducted for the line item service and after standardization of the Medicare payment has been applied.

Table 1 Medicare Part B dictionary

In figure 1 we can notice that the dataset is highly detailed, showing each and every service by the provider, thus we will simplify the dataset grouping the columns. A provider can have the same name as another, therefore we will be grouping as per NPI, since it is unique and will be used as our primary key to link with the other datasets.

	Rndrng_NPI	Rndrng_Prvdr_Last_Org_Name	Rndrng_Prvdr_First_Name	HCPCS_Cd	HCPCS_Desc	Avg_Mdcr_Pymt_Amt
0	1003000126	Enkeshafi	Ardalan	99213	Established patient outpatient visit, total ti...	83.908220
1	1003000126	Enkeshafi	Ardalan	99214	Established patient outpatient visit, total ti...	118.570638
2	1003000126	Enkeshafi	Ardalan	99217	Hospital observation care on day of discharge	61.066923
3	1003000126	Enkeshafi	Ardalan	99220	Hospital observation care, typically 70 minutes	141.442857
4	1003000126	Enkeshafi	Ardalan	99222	Initial hospital inpatient care, typically 50 ...	105.700833
5	1003000126	Enkeshafi	Ardalan	99223	Initial hospital inpatient care, typically 70 ...	170.388889
6	1003000126	Enkeshafi	Ardalan	99226	Subsequent observation care, typically 35 minu...	84.338125
7	1003000126	Enkeshafi	Ardalan	99231	Subsequent hospital inpatient care, typically ...	31.255862
8	1003000126	Enkeshafi	Ardalan	99232	Subsequent hospital inpatient care, typically ...	58.462108
9	1003000126	Enkeshafi	Ardalan	99233	Subsequent hospital inpatient care, typically ...	84.875327
10	1003000126	Enkeshafi	Ardalan	99238	Hospital discharge day management, 30 minutes ...	59.603478
11	1003000126	Enkeshafi	Ardalan	99239	Hospital discharge day management, more than 3...	86.786378
12	1003000126	Enkeshafi	Ardalan	99454	Remote monitoring of physiologic parameters, i...	63.303729
13	1003000126	Enkeshafi	Ardalan	99457	Remote physiologic monitoring treatment manage...	48.005374
14	1003000126	Enkeshafi	Ardalan	99458	Remote physiologic monitoring treatment manage...	38.270000

Figure 1 Medicare dataset

For Categorical features we will keep unique values, i.e., if HCPCS_Cd is more than one is present, it will join the unique entries creating a new entry. For Numerical features, Tot_Benes, Tot_Srvcs and Tot_Bene_Day_Srvcs we will sum and for the other numerical features, we are using weighted average with Tot_Bene_Day_Srvcs as weight. Figure 2 demonstrates how the compilation turns out.

Aggregated dataset shape: 1.123.589 rows, 29 columns

	NPI	Last_Org_Name	First_Name	HCPCS_Cd	HCPCS_Desc	Avg_Mdcr_Pymt_Amt
0	1003000126	Enkeshafi	Ardalan	99213, 99214, 99217, 99220, 99222, 99223, 9922...	Established patient outpatient visit, total ti...	61.441327
1	1003000134	Cibull	Thomas	88304, 88305, 88312, 88313, 88321, 88341, 8834...	Pathology examination of tissue using a micros...	27.620868
2	1003000142	Khalil	Rashid	62323, 64483, 64484, 64490, 64491, 64493, 6449...	Injection of substance into spinal canal of lo...	71.649996
3	1003000423	Velotta	Jennifer	81002, G0101, Q0091	Urinalysis, manual test, Cervical or vaginal c...	31.303448
4	1003000480	Rothchild	Kevin	99202, 99203, 99212, 99213	New patient outpatient visit, total time 15-29...	48.870698
5	1003000530	Semonche	Amanda	81002, 90662, 90670, 90732, 93000, 99213, 9921...	Urinalysis, manual test, Vaccine for influenza...	96.499037
6	1003000597	Kim	Dae	50590, 51102, 51700, 51702, 51705, 51798, 5200...	Shock wave crushing of kidney stones, Aspirati...	73.625755
7	1003000639	Benharash	Peyman	99205	New patient outpatient visit, total time 60-74...	161.090000
8	1003000704	Gatton	Zachary	00142	Anesthesia for lens surgery	118.130500
9	1003000720	Hernandez	Otniel	81003, 99203, 99204, 99205, 99213	Automated urinalysis test, New patient outpati...	68.720287
10	1003000738	Zumwalt	Juliette	20610, 20611, 29823, 29827, 73030, 73562, 9920...	Aspiration and/or injection of large joint or ...	40.744730
11	1003000795	O'neill	Michael	90832	Psychotherapy, 30 minutes	40.528673
12	1003000829	Kochanek	Michelle	97110, 97112, 97140, 97161	Therapeutic exercise to develop strength, endu...	25.300159
13	1003000902	Lohano	Jaivanti	81003, 90662, 90732, 99204, 99213, 99214, 9949...	Automated urinalysis test, Vaccine for influen...	75.950196
14	1003000936	Stellingworth	Mark	36415, 85610, 93000, 93010, 93228, 93270, 9327...	Insertion of needle into vein for collection o...	46.623078

Figure 2 Aggregated Medicare dataset.

2.1.2 LEIE DATASET

LEIE dataset is publicly available at OIG website and is constantly updated.

LEIE dataset shape: 78034 rows, 18 columns

Data dictionary for LEIE:

FIELD VALUE	Description
LASTNAME	Last name
FIRSTNAME	First name
MIDNAME	Mid name
BUSNAME	Business name
GENERAL	Business description
SPECIALTY	Business specialty
UPIN	Unique physician identification number
NPI	National provider identifier
DOB	Date of birth

ADDRESS	Address
CITY	City
STATE	State
ZIP CODE	Zip code
EXCLTYPE	Exclusion type code
EXCLDATE	Exclusion date
REINDATE	Reinstated date
WAIVERDATE	Waiver date
WAIVERSTATE	Waiver state

Table 2 LEIE dictionary.

LEIE dataset comprise 78034 entries. Among these entries, 6713 entries contain NPI. Crossing with Medicare dataset, only 109 yielded (0.14% from LEIE dataset or 0.01% from Medicare dataset) matches with Medicare dataset.

2.1.3 ORDER AND REFERRING DATASET

The Order and Referring (OnR) dataset provide information by the National Provider Identifier (NPI) who are enlisted to being eligible to order and refer in the Medicare program.

Order and Referring dataset shape: 1.785.839 rows, 7 columns.

Data dictionary for Order and Referring:

Field value	Description
NPI	National Provider Identifier (NPI) of the Order and Referring Provider
LAST NAME	Last Name of the Order and Referring Provider
FIRST NAME	First Name of the Order and Referring Provider
PART B	Indicates that provider can refer to Part B

DME	Indicates that provider can order Durable Medical Equipment
HHA	Indicates that provider can refer to Home Health Agency
PMD	Indicates that provider can order Power Mobility Devices

Table 3 Order and Referring dictionary.

Order and Referring dataset is a compilation of the providers and respective eligibility, in this case we will only consider part B eligibility.

	NPI	LAST_NAME	FIRST_NAME	PARTB	DME	HHA	PMD
0	1558467555	.MCINDOE	THOMAS	Y	Y	Y	Y
1	1417051921	A BELLE	N	Y	Y	Y	Y
2	1972040137	A NOVOTNY	ELIZABETH	Y	Y	Y	Y
3	1760465553	A SATTAR	MUHAMMAD	Y	Y	Y	Y
4	1295400745	A'NEAL	BROGAN	Y	Y	N	N
5	1700562584	AAB	BAILEY	Y	Y	Y	N
6	1467482471	AAB	BARRY	Y	Y	Y	N
7	1245971480	AABEDI	ALEXANDER	Y	Y	Y	Y
8	1164905659	AABEL	SAMANTHA	Y	Y	N	N
9	1255630869	AABERG	MAURA	Y	Y	N	N
10	1801093968	AABERG	MELANIE	Y	Y	Y	Y
11	1346991064	AABERG	MICHAEL	Y	Y	Y	Y
12	1588763981	AABERG	RANDAL	Y	Y	Y	Y
13	1194753186	AABERG	THOMAS	Y	Y	Y	Y
14	1891993317	AABIDA	AFEERA	Y	Y	Y	Y
15	1659765857	AABO	MEGHAN	Y	Y	Y	Y
16	1306810700	AABOE	STELLA	Y	Y	Y	Y
17	1164404232	AABY	AAZY	N	Y	N	Y
18	1487775912	AABY	ROYAL	Y	Y	Y	N
19	1508817040	AACH	DOUGLAS	Y	Y	Y	Y

Figure 3 Order and Referring dataset.

Order and Referring dataset comprise 1785839 entries. Among these entries, 55027 appears as N for PARTB. Crossing with Medicare dataset, only 9330 yielded (16.96%

from Order and Referring PARTB as N dataset or 0.83% from Medicare dataset) matches with Medicare dataset.

2.1.4 TARGET VARIABLE

LEIE Dataset and OnR Dataset were used to create the target variable eligibility. Eligibility is a broader concept than fraud, it also considers exclusions due to convictions for program-related fraud as well as offenses such as patient abuse or simply for not being enrolled to Part B but eligible to receive from other federal programs as we could check with OnR dataset.

Additionally, we created flags from Benford's law analyse:

- Largest Growth Test flag;
- Relative Size Factor Test flag; and
- HCPCS flag.

The target will be due to being flagged in Benford's Law analysis or present LEIE or present OnR.

Target Variable:

- **Positive Class (1):** Suspicious providers in the LEIE or in OnR PartB as N or flagged in Benford's Law.
- **Negative Class (0):** Providers not in the LEIE, nor in OnR PartB as N, nor flagged in Benford's Law.

2.2 DATA PROCESSING AND WRANGLING

2.2.1 DATA CLEANING

Cleaning a hyperdimensional dataset might take few steps.

```
df.duplicated().sum()
```

```
0
```

```
df.isna().sum()
```

```
NPI                          0
Last_Org_Name                0
First_Name               64187
MI                      405734
Crdntls                 139077
Gndr                     64187
Ent_Cd                       0
St1                          0
St2                     839121
City                         0
State_Abrvtn                 0
State_FIPS                   0
Zip5                         1
RUCA                       703
RUCA_Desc                  703
Cntry                        0
Type                         0
Mdcr_Prtcptg_Ind             0
HCPCS_Cd                     0
HCPCS_Desc                   0
HCPCS_Drug_Ind               0
Place_Of_Srvc                0
Tot_Benes                    0
Tot_Srvcs                    0
Tot_Bene_Day_Srvcs           0
Avg_Sbmtd_Chrg               0
Avg_Mdcr_Alowd_Amt           0
Avg_Mdcr_Pymt_Amt            0
Avg_Mdcr_Stdzd_Amt           0
fraud                        0
partb_n                      0
EXCLTYPE               1123480
EXCLDATE               1123480
eligibility                  0
dtype: int64
```

Figure 4 Total null values.

In figure 6 shows that we have none duplicated values and 10 columns with null values. EXCLTYPE and EXCLDATE are derived from LEIE dataset, so we can disregard these values. Last_Org_name, First_Name and MI (middle name) information are different ways to check provider identity, therefore we can disregard as we will use NPI, which is unique and does not present null or duplicated values. We will apply the same principle to St2 as we have other columns if address information.

For Gender:

```
len(df[(df.Gndr.isna()) & (df.Ent_Cd == 'O')])
```

```
64187
```

All null values for gender are related to Organization.

```
df.Gndr.fillna('O', inplace = True)
```

Figure 5 Cleaning gender column.

Gender and first name have the same number of null values, hence we checked the possibility that the missing gender are due to being organizations and not individuals. Once that confirmed as per figure 7, created another value: “O”.

For Credentials:

```
df.Crdntls.isna().sum()
```

```
139077
```

```
df[(~df.Crdntls.notna()) & (df.Ent_Cd == 'O')].shape[0]
```

```
64187
```

```
df.loc[df['Ent_Cd'] == 'O', ['Crdntls']] = 'O'
```

```
df.Crdntls.isna().sum()
```

```
74890
```

```
df[(~df.Crdntls.notna()) & (df.Ent_Cd == 'I')].shape[0]
```

```
74890
```

All blanks remaining Crdntls blanks are Individuals

```
df.Crdntls.fillna('I', inplace = True)
```

Figure 6 Cleaning credentials column.

The first assumption is that empty values are due to organizations, once confirmed as per figure 8 it was assigned “O”. Next it was confirmed that the remaining are individuals, hence assigned “I”.

For RUCA:

```
df.RUCA.value_counts()
```

```
1.0     948304
4.0      72957
2.0      29920
7.0      26691
1.1      15218
10.0     11171
5.0       5092
4.1       4201
7.1       1848
3.0       1671
8.0       1367
99.0      1002
7.2        631
6.0        612
9.0        522
10.2       508
2.1        467
10.1       396
10.3       139
5.1        131
8.2         22
8.1         16
Name: RUCA, dtype: int64
```

```
df.RUCA.fillna(0, inplace = True)
```

Figure 7 Cleaning RUCA column.

As per figure 9, we assigned "0" for the blanks.

2.2.2 DIMENSIONALITY REDUCTION

2.2.2.1 DROPPING COLUMNS

Feature	Null count	Type	Relevance
NPI	0	Categorical	Keep
Last_Org_Name	0	Categorical	Drop
First_Name	64187	Categorical	Drop
MI	405734	Categorical	Drop
Crdntls	0	Categorical	Drop
Gndr	0	Categorical	Keep
Ent_Cd	0	Categorical	Drop
St1	0	Categorical	Drop
St2	839121	Categorical	Drop
City	0	Categorical	Drop

State_Abrvtn	0	Categorical	Keep
State_FIPS	0	Categorical	Drop
Zip5	1	Categorical	Drop
RUCA	703	Categorical	Keep
RUCA_Desc	703	Categorical	Drop
Cntry	0	Categorical	Drop
Type	0	Categorical	Keep
Mdcr_Prtcptg_Ind	0	Categorical	Keep
HCPCS_Cd	0	Categorical	Drop
HCPCS_Desc	0	Categorical	Drop
HCPCS_Drug_Ind	0	Categorical	Keep
Place_Of_Srvc	0	Categorical	Keep
Tot_Benes	0	Numerical	Keep
Tot_Srvcs	0	Numerical	Keep
Tot_Bene_Day_Srvcs	0	Numerical	Keep
Avg_Sbmtd_Chrg	0	Numerical	Keep
Avg_Mdcr_Alowd_Amt	0	Numerical	Keep
Avg_Mdcr_Pymt_Amt	0	Numerical	Keep
Avg_Mdcr_Stdzd_Amt	0	Numerical	Keep
fraud	0	Categorical	Drop
partb_n	0	Categorical	Drop
EXCLTYPE	1123480	Categorical	Drop
EXCLDATE	1123480	Date	Drop
eligibility	0	Categorical	Keep

Table 4 Feature relevance.

In table 5 features are separated per categories:

Blue – ID

ID won't contribute with the model, but we kept NPI as reference for now.

Red – ID classification

Credentials has too many unique values, therefore disregard due to high cardinality. Ent_Cd provide almost the same information as Gender, hence we keep only Gender.

Yellow – Geolocation

St1, St2, City, State_FIPS, Zip5 and Cntry are implicit in State_ Abrvtn. RUCA_Desc is implicit in RUCA, hence we keep State_ Abrvtn and RUCA, that might bring a different perception for the model.

Grey – Medical service related

This is one of the most relevant categories, which we will drop only HCPCS_Cd and HCPCS_Desc that apparently is a more detailed version of Type column.

White – FFS

Another relevant categories since we are trying to predict eligibility based on FFS data.

Green – Target feature

Feature "fraud" is named after matching LEIE dataset, partb_n is named after matching OnR dataset and eligibility is the merged columns, hence we drop the first two. EXCLTYPE and EXCLDATE are does not look like relevant data since almost all the values are null.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1123589 entries, 0 to 1123588
Data columns (total 19 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   NPI                1123589 non-null  int64
 1   Crdntls            1123589 non-null  object
 2   Gndr               1123589 non-null  object
 3   Ent_Cd             1123589 non-null  object
 4   State_Abrvtn       1123589 non-null  object
 5   RUCA               1123589 non-null  float64
 6   Type               1123589 non-null  object
 7   Mdcr_Prtcptg_Ind   1123589 non-null  object
 8   HCPCS_Cd           1123589 non-null  object
 9   HCPCS_Drug_Ind     1123589 non-null  object
 10  Place_Of_Srvc      1123589 non-null  object
 11  Tot_Benes          1123589 non-null  int64
 12  Tot_Srvcs          1123589 non-null  float64
 13  Tot_Bene_Day_Srvcs 1123589 non-null  int64
 14  Avg_Sbmtd_Chrg     1123589 non-null  float64
 15  Avg_Mdcr_Alowd_Amt 1123589 non-null  float64
 16  Avg_Mdcr_Pymt_Amt  1123589 non-null  float64
 17  Avg_Mdcr_Stdzd_Amt 1123589 non-null  float64
 18  eligibility        1123589 non-null  int64
dtypes: float64(6), int64(4), object(9)
memory usage: 162.9+ MB
```

Figure 8 Dataset after dimensionality reduction.

2.2.2.2 CHI SQUARE

For further feature selection, it was implemented Chi Squared. We already selected 7 features, but we want to further reduce the dimensionality selecting the 5 most relevant.

```
df.RUCA = df.RUCA.astype(object)
```

```
cat_col = ['Gndr', 'State_Abrvtn', 'RUCA','Type',
    'Mdcr_Prtcptg_Ind', 'HCPCS_Drug_Ind', 'Place_Of_Srvc']
```

```
X = df[cat_col]
y = df.eligibility
```

```
X.describe()
```

	Gndr	Ent_Cd	State_Abrvtn	RUCA	Type	Mdcr_Prtcptg_Ind	HCPCS_Drug_Ind	Place_Of_Srvc
count	1123589	1123589	1123589	1123589.0	1123589	1123589	1123589	1123589
unique	3	2	61	23.0	103	4	4	4
top	M	I	CA	1.0	Nurse Practitioner	Y	N	O
freq	559187	1059402	87424	948304.0	149911	1122070	901567	561001

```
label_encoder = LabelEncoder()
for col in cat_col:
    X[col] = label_encoder.fit_transform(X[col])
```

Figure 9 Converting categorical features.

Figure 11 shows the columns selected for categorical feature selection and transforming the data.

```
chi2_stats, p_values = chi2(X, y)

results_df = pd.DataFrame({'Feature': cat_col, 'Chi-squared': chi2_stats, 'p-value': p_values})

print(results_df)
```

```
            Feature  Chi-squared       p-value
0              Gndr    74.781743  5.257365e-18
1      State_Abrvtn   104.697243  1.422989e-24
2              RUCA    23.407752  1.310494e-06
3              Type  1806.206803  0.000000e+00
4   Mdcr_Prtcptg_Ind     0.008807  9.252312e-01
5     HCPCS_Drug_Ind    16.911569  3.916217e-05
6      Place_Of_Srvc   144.343798  2.988307e-33
```

```
X_new = SelectKBest(score_func=chi2, k=5).fit_transform(X, y)
```

```
selector = SelectKBest(score_func=chi2, k=5)
X_new = selector.fit_transform(X, y)

selected_feature_indices = selector.get_support(indices=True)

selected_features = X.columns[selected_feature_indices]

print("Selected Features:")
print(selected_features)
```

```
Selected Features:
Index(['Gndr', 'State_Abrvtn', 'RUCA', 'Type', 'Place_Of_Srvc'], dtype='object')
```

Figure 10 Chi Squared results.

The second step is running Chi Squared and selecting the best values. As we can see in the first result shown in Figure 12, Type, Ent_cd, Place_Of_Srvc, State_Abrvtn and Gndr had the highest Chi-squared values with p-values < 0.05. We used SelectKBest to confirm our interpretation.

2.2.2.3 MULTICOLLINEARITY

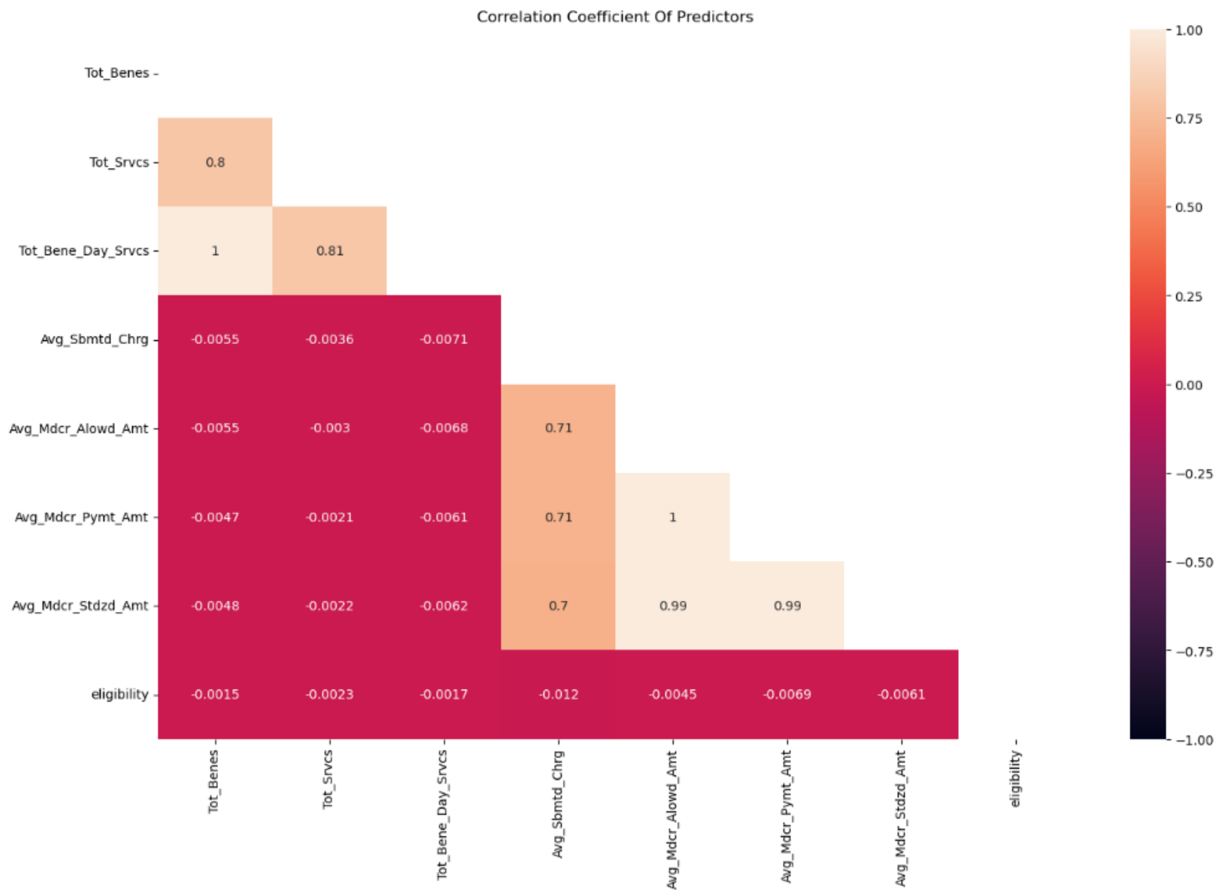


Figure 11 Numerical features correlation.

There is low to no correlation between the independent features and target feature, however it shows collinearity between some independent features. To deal with collinearity, was verified Variance Inflation Factor (VIF) and four iterations was performed until we reach values lower than 5 as per figures 14-17.

```python
X = data_num.drop(columns=['eligibility'])
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_data
```

	Feature	VIF
0	Tot_Benes	113.146093
1	Tot_Srvcs	2.909676
2	Tot_Bene_Day_Srvcs	118.421456
3	Avg_Sbmtd_Chrg	2.009628
4	Avg_Mdcr_Alowd_Amt	269.597469
5	Avg_Mdcr_Pymt_Amt	340.426661
6	Avg_Mdcr_Stdzd_Amt	86.906657

Figure 12 VIF first iteration.

```python
X = data_num.drop(columns=['eligibility','Tot_Benes'])
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_data
```

	Feature	VIF
0	Tot_Srvcs	2.870310
1	Tot_Bene_Day_Srvcs	2.870301
2	Avg_Sbmtd_Chrg	2.009421
3	Avg_Mdcr_Alowd_Amt	269.591667
4	Avg_Mdcr_Pymt_Amt	340.422573
5	Avg_Mdcr_Stdzd_Amt	86.906383

Figure 13 VIF second iteration.

```python
X = data_num.drop(columns=['eligibility','Tot_Benes','Avg_Mdcr_Alowd_Amt'])
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_data
```

	Feature	VIF
0	Tot_Srvcs	2.870179
1	Tot_Bene_Day_Srvcs	2.870291
2	Avg_Sbmtd_Chrg	2.009356
3	Avg_Mdcr_Pymt_Amt	88.346743
4	Avg_Mdcr_Stdzd_Amt	86.635528

Figure 14 VIF third iteration.

```python
X = data_num.drop(columns=['eligibility','Tot_Benes', 'Avg_Mdcr_Alowd_Amt', 'Avg_Mdcr_Pymt_Amt'])
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_data
```

	Feature	VIF
0	Tot_Srvcs	2.870176
1	Tot_Bene_Day_Srvcs	2.870291
2	Avg_Sbmtd_Chrg	1.967601
3	Avg_Mdcr_Stdzd_Amt	1.967593

Figure 15 VIF fourth iteration.

If we check the dictionary, we can see that Tot_Benes_Day_Srvcs is related to Tot_Benes and Avg_Mdcr,_Stdzd_Amt is related to Avg_Mdcr_Alowd_Amt and Avg_Mdcr_Pymt_Amt.

2.2.3 FEATURES TRANSFORMATION

2.2.3.1 STANDARDIZATION

Next, we need to transform all the data to suitable forms in which our machine learning models will be able to access and interpretate. Standardization process was chosen for numerical features transformation as per following figure 18.

```python
numerical_columns = ['Tot_Bene_Day_Srvcs', 'Tot_Srvcs','Avg_Sbmtd_Chrg', 'Avg_Mdcr_Stdzd_Amt']
```

```python
data_num[numerical_columns] = StandardScaler().fit_transform(data_num[numerical_columns])
```

Figure 16 Numerical features transformation.

2.2.3.2 LABEL ENCODING

For categorical features, we will try two approaches: label encoding and one-hot encoding. Label encoding is a process of assigning numerical labels to categorical data values, which might give an implicit ordinality that is not ideally for our case. Figure 19 demonstrate label encoding transformation.

```python
df_label = df.copy()
```

```python
label_encoder = LabelEncoder()
for col in categorical_columns:
    df_label[col] = label_encoder.fit_transform(df[col])
```

```python
df_label[numerical_columns] = StandardScaler().fit_transform(df_label[numerical_columns])
```

```python
df_label = df_label[categorical_columns+numerical_columns+target]
```

Figure 17 Label encoding transformation.

2.2.3.3 ONE-HOT ENCODING

The other categorical feature transformation selected for screening is one-hot encoding. One-hot encoding convert a categorical variable with k distinct categories into k separate binary features, each representing one category. Each binary feature is converted to 1 or 0.

Figure 20 demonstrate one-hot encoding transformation.

```python
df_reduced = df[categorical_columns+numerical_columns+target]
```

```python
df_reduced.shape
```

```
(1123589, 10)
```

```python
encoder = OneHotEncoder(sparse = False)
```

```python
encoder.fit(df_reduced[categorical_columns])
df_reduced_encoded = encoder.transform(df_reduced[categorical_columns])
df_reduced_encoded_df = pd.DataFrame(df_reduced_encoded, columns=encoder.get_feature_names(categorical_columns))
df_reduced_encoded_df.index = df_reduced.index
```

```python
df_reduced_encoded_df.shape
```

```
(1123589, 194)
```

```python
df_combined = df_reduced_encoded_df.join(data_num)
```

```python
df_combined = df_combined.drop(columns=['Tot_Bene_Day_Srvcs', 'Avg_Mdcr_Alowd_Amt', 'Avg_Mdcr_Pymt_Amt'])
```

Figure 18 One-hot encoding transformation.

2.3 IMBALANCE DATA

Imbalanced dataset refers to a situation where the distribution of the target variable (in this case, whether a provider is eligible or not) is significantly skewed. In other words, one class has much fewer instances compared to the other class, consequently, pose challenges for machine learning models because models may become biased toward the majority class. In fraud detection, where the occurrence of fraud is typically rare, a model trained on imbalanced data might struggle to accurately identify instances of fraud.

Traditional machine learning algorithms can be biased towards the majority class, leading to high accuracy but poor performance in identifying the minority class (fraud). The model might be too conservative and tend to predict non-fraudulent cases more frequently. Strategies to address imbalanced data include resampling techniques (oversampling the minority class or under sampling the majority class), using different evaluation metrics (precision, recall, F1 score), or employing specialized algorithms designed for imbalanced datasets.

2.3.1 SAMPLING

We used two sampling methods: oversampling and under sampling. Oversampling is a method that adds samples to the minority class, while under sampling balances the dataset by removing samples from the majority class.

2.3.1.1 OVER SAMPLING

We used the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to oversample the minority class in your training data (X_train, y_train).

```
y_train_val.value_counts()
```

```
0    44553
1      390
Name: eligibility, dtype: int64
```

```
smote = SMOTE(random_state=42)
X_train_smote_val, y_train_smote_val = smote.fit_resample(X_train_val, y_train_val)
```

```
y_train_smote_val.value_counts()
```

```
0    44553
1    44553
Name: eligibility, dtype: int64
```

Figure 19 SMOTE sampling.

Figure 21 shows the outcome when using SMOTE: we are creating entries for the imbalanced data to match with the one in majority.

2.3.1.2 UNDER SAMPLING

For under sampling, it was retained the same amount of the eligibility label rows for the majority label, thus ensuring that we have a balanced dataset but with the risk of information loss.

```
df_under = pd.concat([X_train_val, y_train_val], axis=1)
```

```
df_under.eligibility.value_counts()
```

```
0    44553
1      390
Name: eligibility, dtype: int64
```

```
fraud_df = df_under.loc[df['eligibility'] == 1]
non_fraud_df = df_under.loc[df['eligibility'] == 0][:390]
```

```
normal_distributed_df = pd.concat([fraud_df, non_fraud_df])
# Shuffle dataframe rows
df_new = normal_distributed_df.sample(frac=1, random_state=42)
# split out validation dataset for the end
y_train_under= df_new["eligibility"]
X_train_under = df_new.loc[:, df.columns != 'eligibility']
```

```
df_new.eligibility.value_counts()
```

```
0    390
1    390
Name: eligibility, dtype: int64
```

Figure 20 Random under sampling.

Figure 22 shows the outcome when using under sampling: we are excluding entries from the majority label to match with eligibility label.

2.3.2 MODELS METRIC

In fraud detection, the interpretation of classification metrics is crucial for understanding how well a model performs in identifying fraudulent transactions. As mentioned before, it is also important to have low False Positive outcome and since we are running machine learning model screening with a highly imbalanced dataset, it is key to properly define our metrics priority.

A high recall means that the model can correctly identify “not eligible” instances with few false negatives, but not choosing precision might yield lots of false positive cases which is not ideal. Usually, accuracy is the metric to go with, but for our project the aim is to identify the most cases labeled as “1”, i.e., not eligible, therefore we will use a multimetric system with recall as the main metric followed by accuracy. The goal is to achieve both metrics above 0.60.

3 RESULTS

3.1 How real-world data behave when analyzed with forensic accounting tools from Benford's Law?

In Benford's Law it is expected frequencies of digits to behave in a certain way. The test can be done in first order or second order, analyzing any position or combination of digits. In our project, we explored forensic analytics tools present in the book from Mark J. Nigrini, Forensic Analytics: Methods and Techniques for Forensic Accounting Investigations:

- First Order First Digit Test,
- First Order Second Digit Test,
- First Order First-Two Digit Test,
- First Order Last Two Digit,
- Second Order First-Two Digits Test,
- Summation Test,
- Largest Growth Test,
- Relative Size Factor Test.

We ran the tests using average of the charges that the provider submitted for the service (Avg_Sbmtd_Chrg) from the original Medicare dataset (9886177 entries). To assess conformity with Benford's Law, we chose the mean absolute deviation (mad) test:

$$\text{Mean Absolute Deviation} = \frac{\sum_{i=1}^{K} |AP - EP|}{K}$$

where EP denotes the expected proportion, AP the actual proportion, and K represents the number of bins.

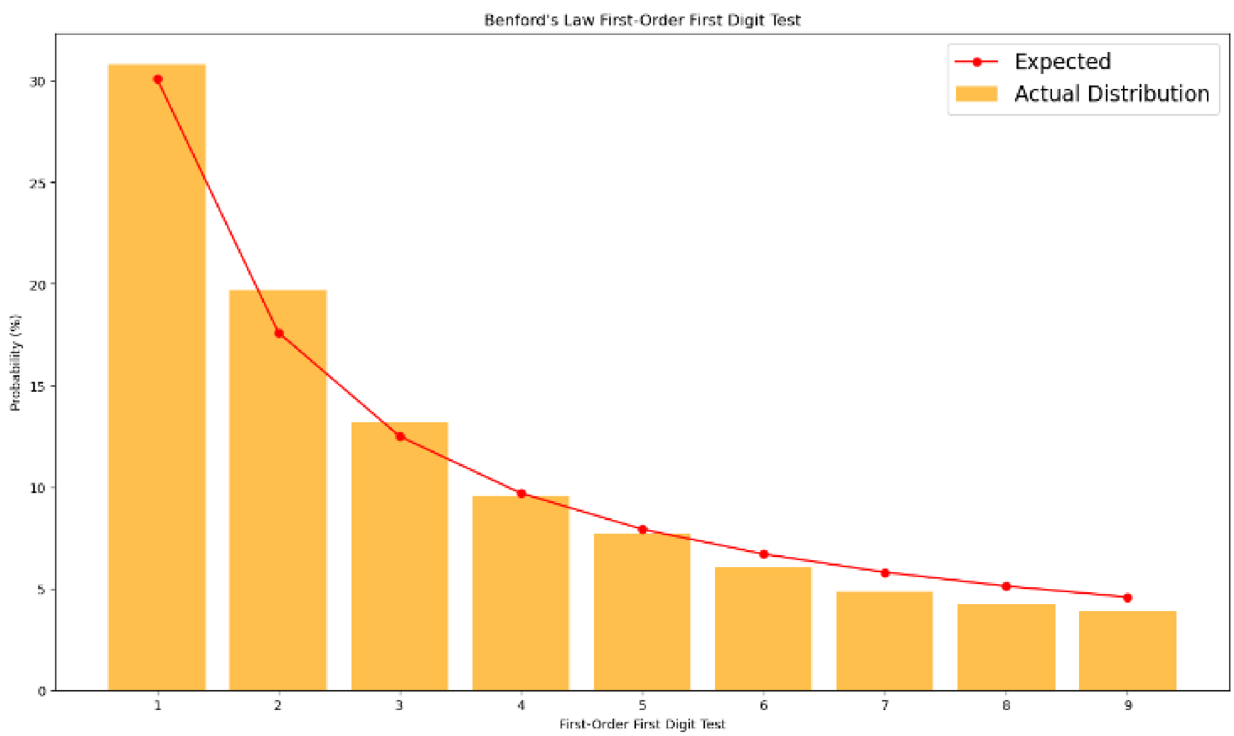
According to the author:

First-Two Digits MAD Range	Conclusion
0.0000 to 0.0012	Close conformity
0.0012 to 0.0018	Acceptable conformity
0.0018 to 0.0022	Marginally acceptable conformity
Above 0.0022	Nonconformity

Table 5 MAD range

The authors tested several datasets and reached the above table. Despite the MAD values being developed for First-Two Digits Test, for simplicity we will apply for all the others tests.

3.1.1 First Order First Digit Test

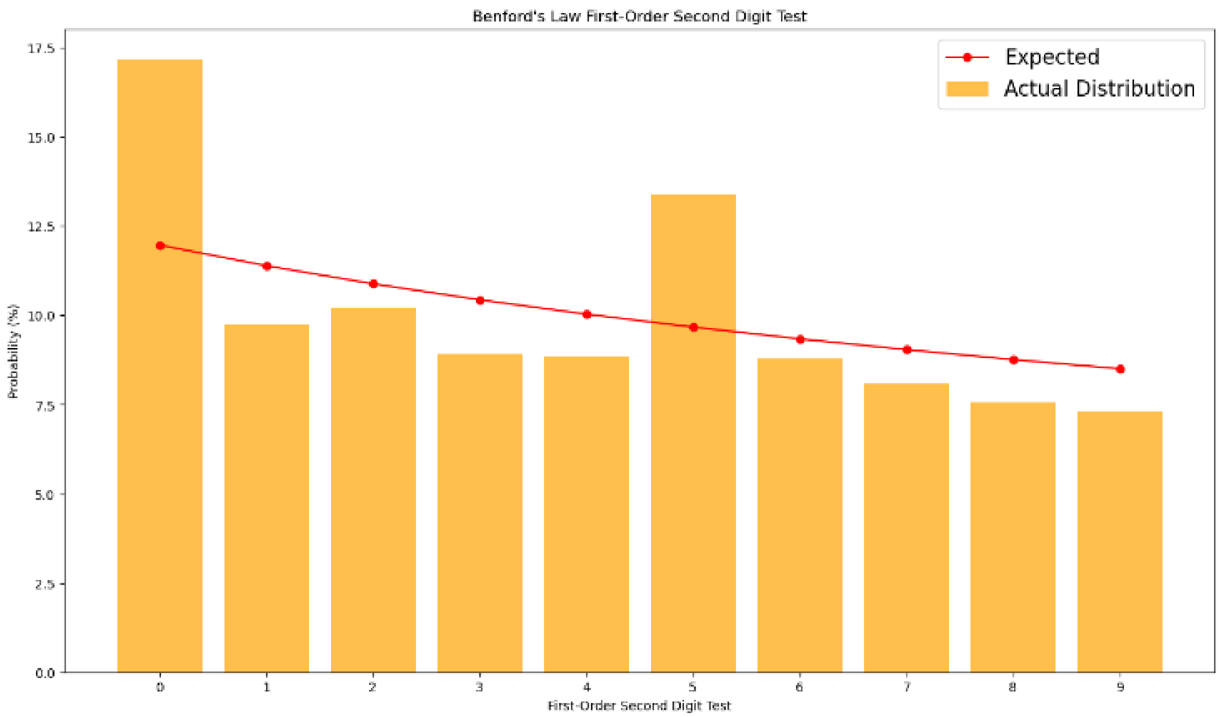



```
MAD: 0.0078 - Nonconformity
```

Figure 21 First Order First Digit Test

In our first test, actual distribution behaved close to expected, yet MAD yielded nonconformity. This first test shows that we have a high distribution of first digit 2. Since the first digit already yielded nonconformity, we explored further tests.

3.1.2 First Order Second Digit Test

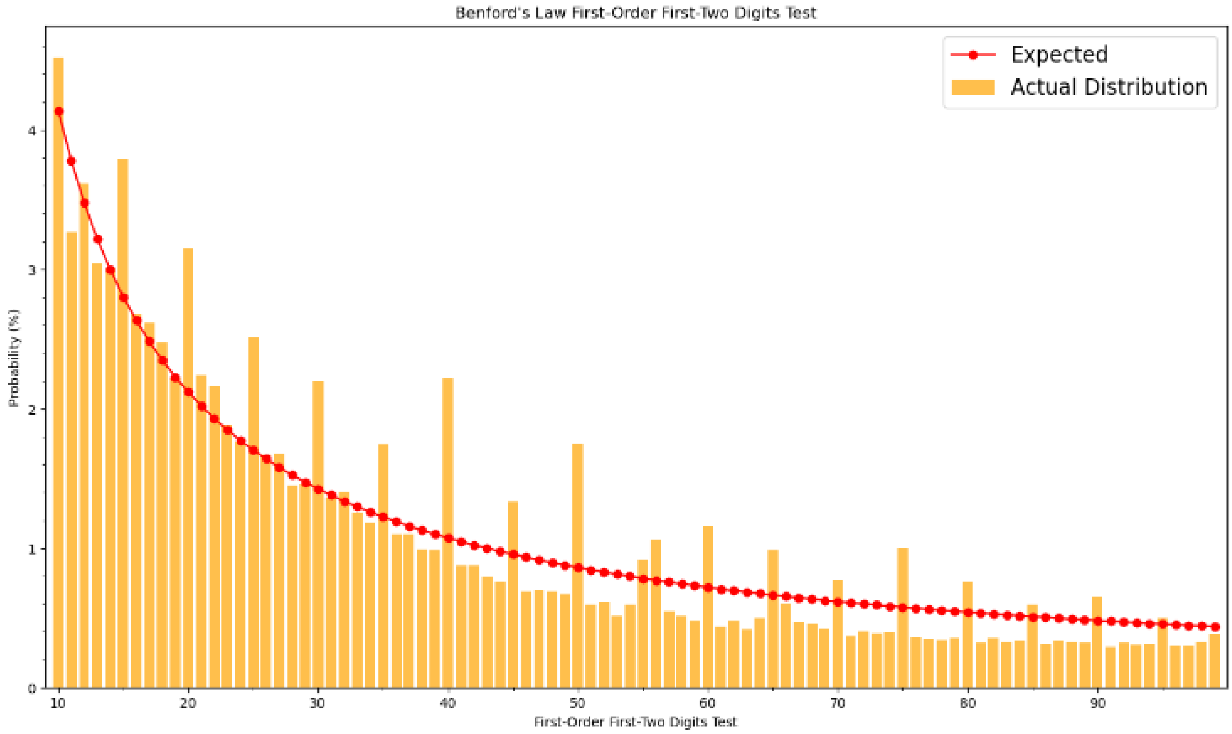


`MAD: 0.0178 - Nonconformity`

Figure 22 First Order Second Digit Test

For the second digit, clearly, we have high frequency of second digit 0 and 5. Average submitted charge that is less than 100 amount for 35% of the entries, therefore might indicate that providers usually submit values that end as 0 or 5.

3.1.3 First Order First-Two Digit Test




```
MAD: 0.0023 - Nonconformity
```

Figure 23 First Order First-Two Digit Test

As expected from first and second digit test, when we ran first-two digit test it is clear that values that end with 0 and 5 are higher, but oddly 56 is a number above expected and even above than 55. For now, we will just take note about this fact since we lack expertise to further explore or discuss reasons why this behavior is present in Medicare dataset.

3.1.4 First Order Last Two Digit

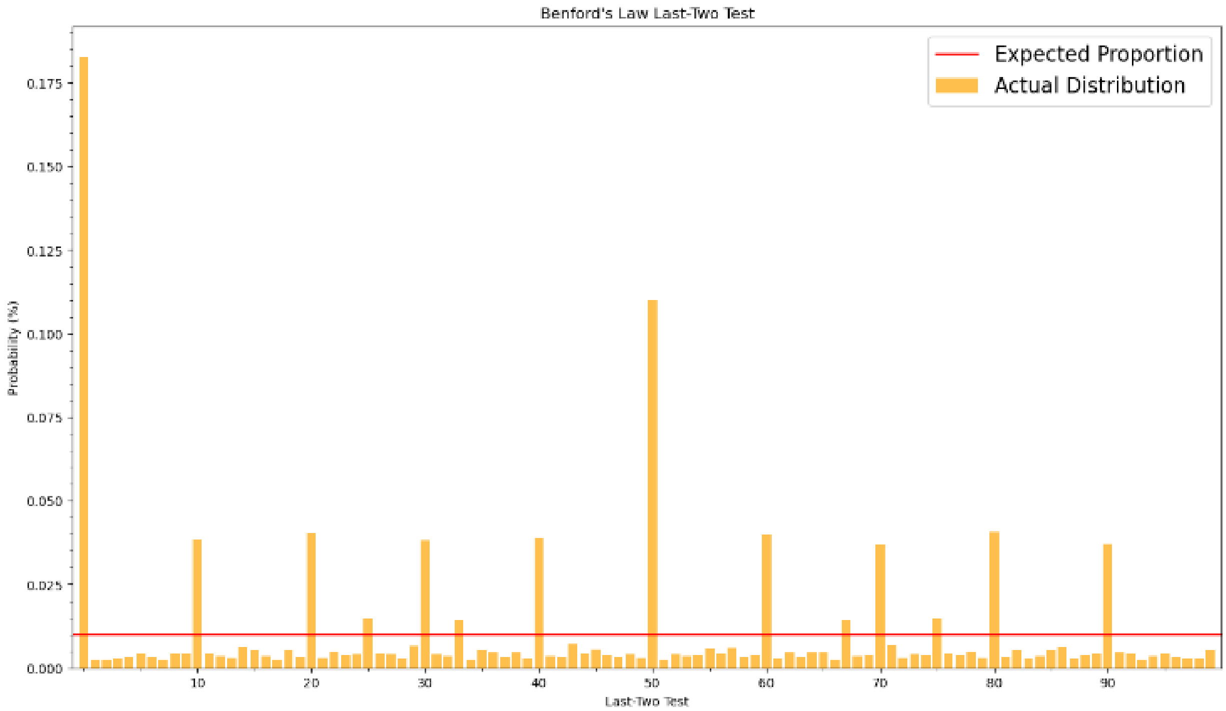
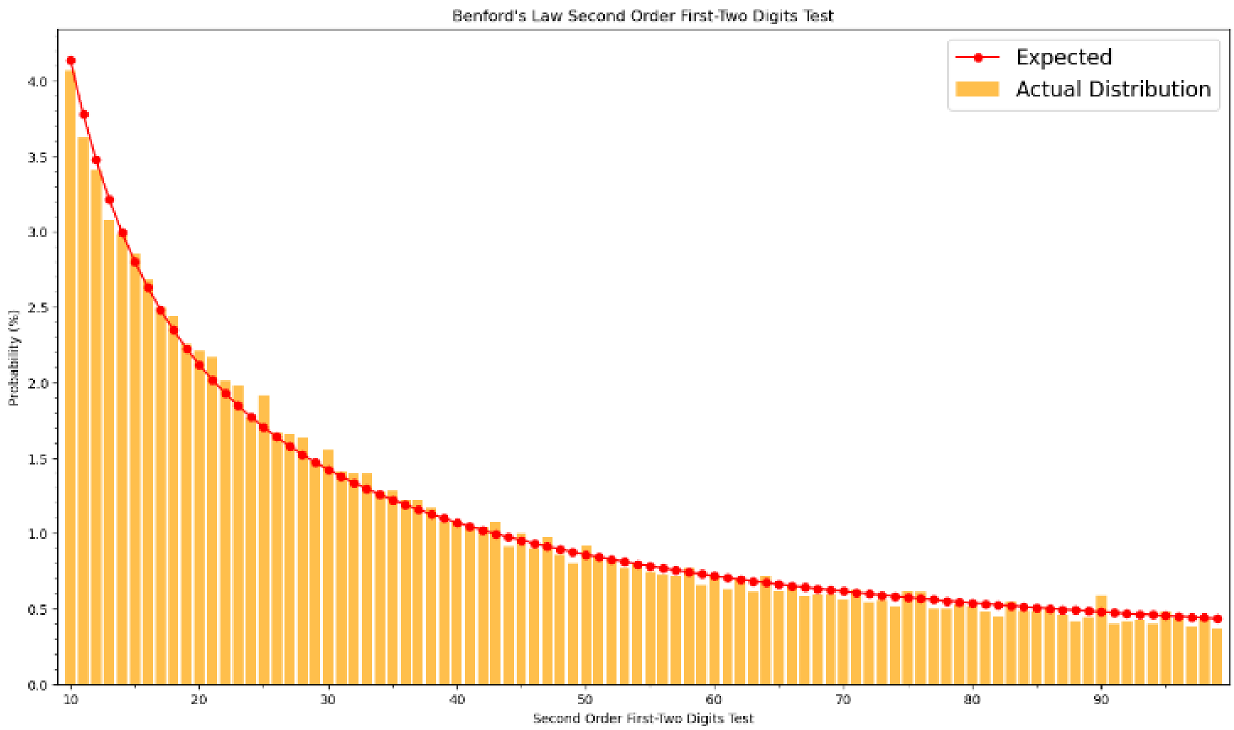


Figure 24 First Order Last Two Digit

For this test, the expected proportion is set at 0.01 for all values. We have a huge discrepancy for 00 and 50, but the others values ending with 0 also present spikes above expected.

3.1.5 Second Order First-Two Digits Test




```
MAD: 0.0005 - Close conformity
```

Figure 25 Second Order First-Two Digits Test

Second order is calculated by the difference between the current entry and previous entry: y_n-y_{n-1}. For this test, the results were very close to expected, with few spikes between 20-30 and at 90.

3.1.6 Summation Test

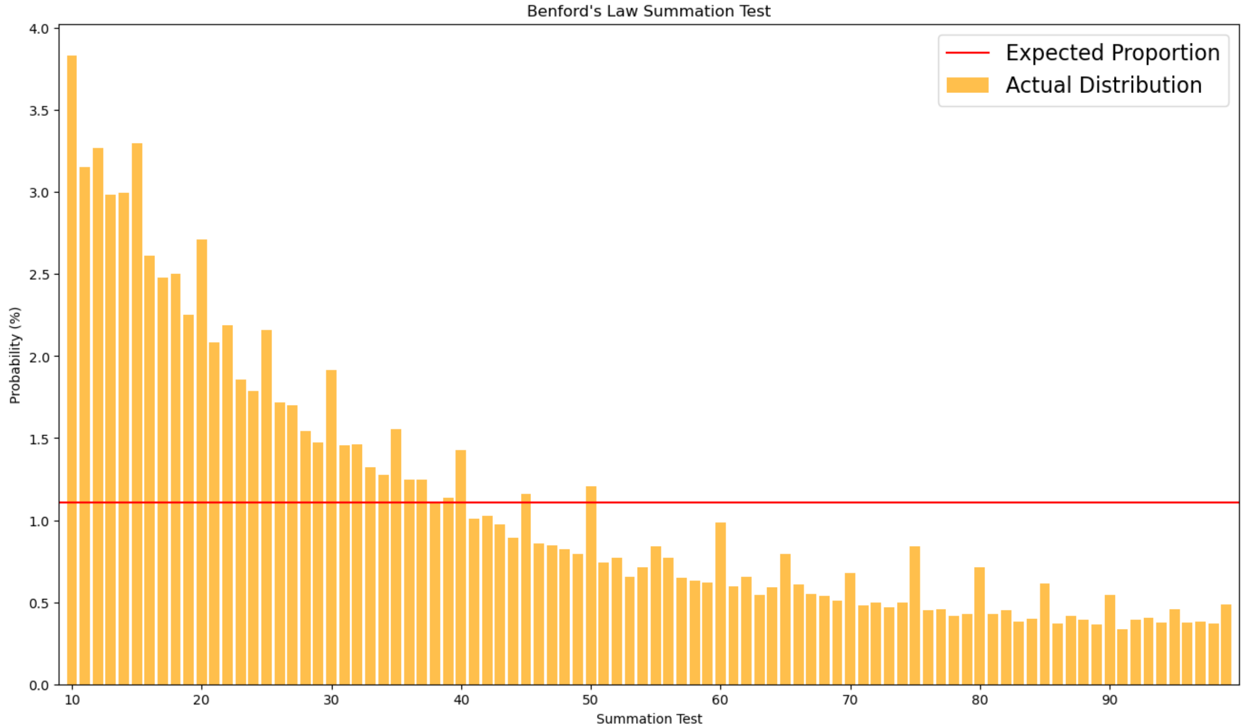


Figure 26 Summation Test

For summation test, as per first-two digit we sum all the values corresponding to the first-two digit and divide by the total sum of all digits. The expected proportion is set at 0.01 for all values.

3.2 In case we have discrepancies between real-world data and expected Benford's Law, how can we leverage this information to flag transactions as suspicious?

Our reference has other testing methods such as Largest Growth test and Relative Size Factor test. For our problem, we made a few adjustments to reflect in our data. Also, with this tests we were inspired to further do a bivariate test according to HCPCS code and average submitted charge by NPI.

3.2.1 Largest Growth Test

	Rndrng_NPI	Sum_2021	Tot_Benes_2021	Sum_2020	Tot_Benes_2020	Sum_growth	Tot_bene_growth	Sum_bene_ratio	ratio_flag
0	1538144910	98401.906275	10948038.0	97027.700598	10342594.0	1.014163	1.058539	0.958078	0
1	1891731626	108890.330026	5586675.0	110824.887552	5467219.0	0.982544	1.021849	0.961535	0
2	1932145778	105890.329231	5296381.0	97742.967041	4111244.0	1.083355	1.288267	0.840940	0
3	1063497451	98077.502440	6877772.0	95401.872824	6108559.0	1.028046	1.125924	0.913069	0
4	1366479099	100922.779814	4854261.0	96616.018640	4414351.0	1.044576	1.099655	0.949913	0
...	...	...	...	...	...	...	...	...	...
1123582	1215033550	333.286385	17.0	350.000000	41.0	0.952247	0.414634	2.296595	1
1123583	1902162688	191.000000	37.0	332.000000	41.0	0.575301	0.902439	0.637496	0
1123584	1528180171	295.000000	82.0	295.000000	36.0	1.000000	2.277778	0.439024	0
1123585	1528180197	993.750000	16.0	1671.250000	12.0	0.594615	1.333333	0.445961	0
1123587	1528180361	256.000000	16.0	320.000000	25.0	0.800000	0.640000	1.250000	0

996091 rows × 9 columns

Figure 27 Largest Growth Test

In this test we explore the growth of the sum charged. Since we want to identify suspicious growth, i.e., we did the sum growth ratio against total beneficiaries' growth. If this ratio was above 2, we raised a flag indicating suspicious values submitted.

3.2.2 Relative Size Factor Test

	max_1	max_2	rsf	rsf_flag
0	1165.704744	1122.043654	1.038912	0
1	2762.000000	2504.675000	1.102738	0
2	3082.260278	2762.000000	1.115952	0
3	1160.750857	1122.000000	1.034537	0
4	3050.000000	2990.000000	1.020067	0
...	...	...	...	...
1123582	333.286385	0.000000	0.000000	0
1123583	191.000000	0.000000	0.000000	0
1123584	295.000000	0.000000	0.000000	0
1123585	993.750000	0.000000	0.000000	0
1123587	256.000000	0.000000	0.000000	0

996091 rows × 4 columns

Figure 28 Relative Size Factor Test

RSF divide the max value submitted against the second highest value. If this yield a high ratio, in this case if the max value is at least double than the second highest , we raised a suspicious flag as well.

3.2.3 HCPCS Test

	Rndrng_NPI	HCPCS_Cd	Avg_Sbmtd_Chrg	Avg_Sbmtd_Chrg_HCPCS	HCPCS_ratio	HCPCS_flag
0	1003000126	99213	125.000000	150.000000	0.833333	0
1	1003000126	99214	173.829787	220.173102	0.789514	0
2	1003000126	99217	257.620513	187.363636	1.374976	0
3	1003000126	99220	1192.656191	471.000000	2.532179	0
4	1003000126	99222	319.666667	309.000000	1.034520	0
...	...	...	...	...	...	...
9886172	1992999825	99214	291.000000	220.173102	1.321687	0
9886173	1992999874	99223	699.117647	465.050000	1.503317	0
9886174	1992999874	99232	249.467181	172.000000	1.450391	0
9886175	1992999874	99233	355.699670	253.000000	1.405928	0
9886176	1992999874	99239	368.638554	269.000000	1.370404	0

9886177 rows × 6 columns

Figure 29 HCPCS Test

For this test, we extracted the median average submitted charge for each HCPCS code (HCPCS_Cd). Then we individually compared through a ratio between these values with the ones that each provider submitted. If the average submitted charge HCPCS ratio was higher than 3, we raise a flag.

3.2.4 Target Variable

Our target variable is now improved:

- eligibility due to appearing in LEIE or OnR dataset.
- Largest Growth Test flag.
- Relative Size Factor Test flag.
- HCPCS Test flag.

From 0.84% of not eligible providers, with the criteria selected for all three test flags, we jumped to 24.25% of providers being flagged. It seems a high value, but for the sake of simplicity, we kept these values to reduce the imbalance from our dataset.

3.3 Can we create deep learning feedforward neural network able to perform better than our benchmarked supervised learning model?

3.3.1 Supervised Machine Learning

For our previous model using Gradient Boosting Classifier using only eligibility from LEIE and OnR dataset as target and Under Sampling, we have the results below:

	precision	recall	f1-score	support
0	1.00	0.59	0.74	222852
1	0.02	0.76	0.03	1866
accuracy			0.59	224718
macro avg	0.51	0.67	0.39	224718
weighted avg	0.99	0.59	0.74	224718

Figure 30 GBM results

Now, when we run the same model using additional target flags, we have a reduced recall, but better accuracy.

	precision	recall	f1-score	support
0	0.86	0.72	0.79	127661
1	0.42	0.63	0.51	40878
accuracy			0.70	168539
macro avg	0.64	0.68	0.65	168539
weighted avg	0.75	0.70	0.72	168539

Figure 31 GBM with flags results

3.3.2 Deep neural network

For the feedforward neural network, we tested several models until reaching our final model:

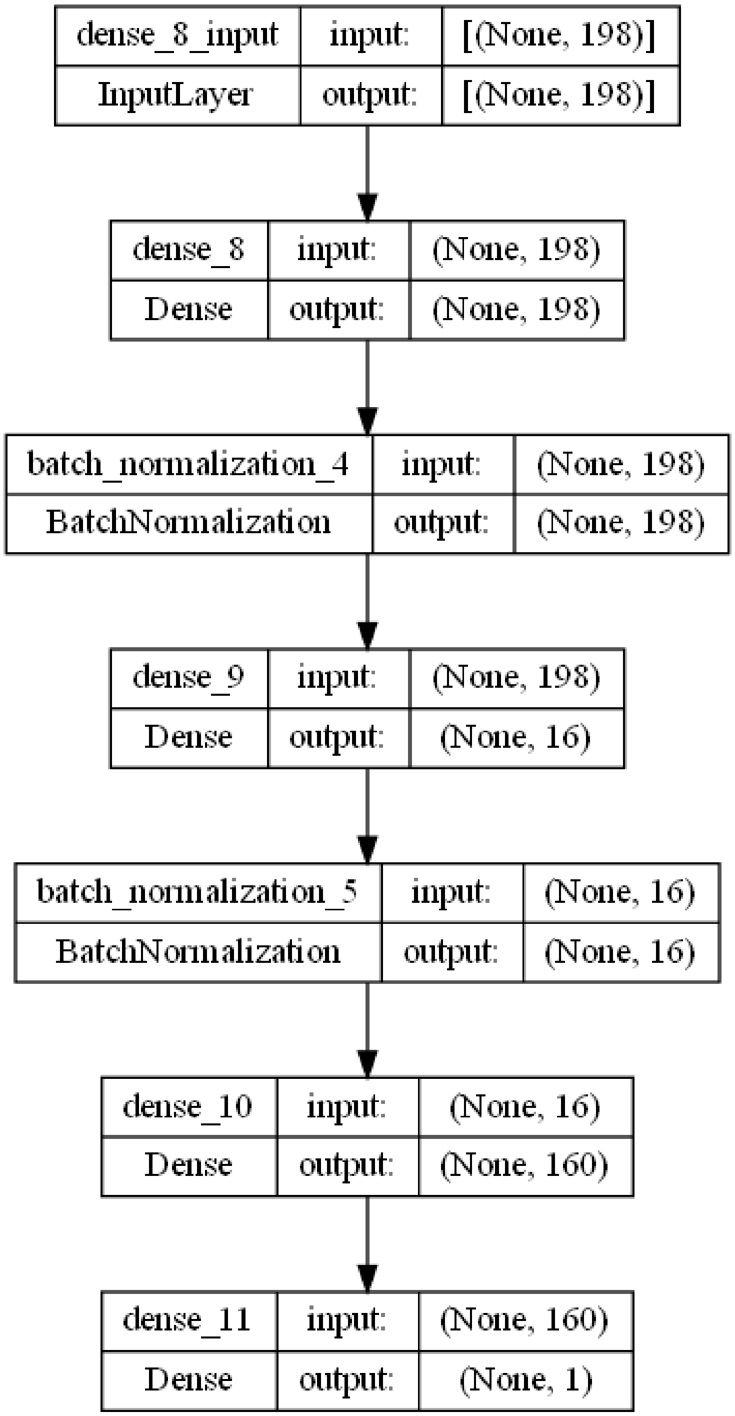


Figure 32 FNN model

This model yields the best results when running with over sampling (SMOTE). To compare both supervised machine learning and deep learning, we ran the model with both target variables, only with eligibility and using additional flags.

The results using only eligibility wasn't good, with low accuracy and recall.

```
48745/48745 [==============================] - 91s 2ms/step - loss: 0.5218 - accuracy: 0.7249 - recall_3: 0.8065 - val_loss: 0.
5102 - val_accuracy: 0.6645 - val_recall_3: 0.6163
5267/5267 [==============================] - 6s 1ms/step - loss: 0.8324 - accuracy: 0.5412 - recall_2: 0.3750
Test Accuracy: 0.5412
Test Recall: 0.3750
```

Figure 33 FNN results

However, using the additional flags, in comparison with supervised model, the neural network outperformed.

```
37235/37235 [==============================] - 71s 2ms/step - loss: 0.6028 - accuracy: 0.6714 - recall_2: 0.6328 - val_loss: 0.
6710 - val_accuracy: 0.6387 - val_recall_2: 0.7028
5267/5267 [==============================] - 6s 1ms/step - loss: 0.6729 - accuracy: 0.6374 - recall_2: 0.7016
Test Accuracy: 0.6374
Test Recall: 0.7016
```

Figure 34 FNN with flags results

In this first approach, we had satisfactory results, but in future work we could try to improve using Keras Tuner.

4 DISCUSSION

How real-world data behave when analyzed with forensic accounting tools Benford's Law?

Fraud detection is a broad subject without pre-defined set rules to be followed, however we can be guided by analytics tools, in this case, forensic accounting analytical tools to guide us until we define premises specific for the collected data.
Benford's law can be used in the exploratory data analysis (EDA) process and lead us to insightful facts. Also, Benford's law set a preposition that data is presented in a certain way, which in a natural order it should, despite a few spikes in certain values due to intrinsic characteristic of the data. For example, it is expected that bills charges have higher amounts of values ending in 0, such as 10, 50 and 90.

Test	Conclusion
First Order First Digit Test	Nonconformity
First Order Second Digit Test	Nonconformity
First Order First-Two Digit Test	Nonconformity
First Order Last Two Digit	Nonconformity
Second Order First-Two Digits Test	Close conformity
Summation Test	Nonconformity

Table 6 Benford's Law test results

Out of six tests, only one had close conformity and the others nonconformity. Which induce us to explore more our data.

In case we have discrepancies between real-world data and expected Benford's Law, how can we leverage this information to flag transactions as suspicious?

Leveraging the information diverted from Benford's Law tests, we figured that we could create our own flags to signalize if something seems abnormal. That being sad, we created three criteria to define if an average submitted charge should be flagged as suspicious.

The first one, we need historical data to confront with the last data available. In this case, we compared 2021 and 2020 data. If the charges growth rate in the period is too high, not being justifiable by the increase of new beneficiaries, we would raise a flag, but please notice that price adjustment, for example due to inflation could occur, hence this information alone won't define as fraud, but a flag to keep an eye on it.

RSF test is used to raise flags in possible errors, in this case, if a max value is way to high in contrast to the second max value, this might raise an alert that an error is occurring. For example, if a person usually pays 700-900 for credit card bill, but for this month the bill is 3000$ is due to a flight ticket bought for vacation, the bank should emit an alert to the customer identifying an anormal value. The same logic can be applied for our dataset, hence we set a criterion that in case RSF ratio is high, we should investigate to understand the reason.

Our last test was a simple test to compare the median values charged for each HCPCS code used to identify the specific medical service furnished by the provider. Hence if a provider charges more than double of the median price of the specific HCPCS code, we should investigate why is occurring overcharging for the same services provided by others.

Can we create a deep learning feedforward neural network able to perform better than our benchmarked supervised learning model?

Using only the information available before, our supervised learning model greatly outperformed FNN. When we run the new data, the supervised model has better accuracy and lower recall. FNN with the flags had better recall and an acceptable accuracy.

	Accuracy	Recall
GBM	0.59	0.76
GBM flags	0.70	0.63
FNN	0.54	0.37
FNN flags	0.64	0.70

Table 7 Models comparison

In the end, it is debatable which model is better. Since we trained with the whole dataset, both GB and FNN consume a lot of memory and are complex, however analyzing only the metrics, we could say that FNN is slightly better.

5 CONCLUSION

Forensic accounting tools improved our vision on how to assess the data and gave us insightful visions. We could explore deeper why almost all Benford`s Law tests were not conforming and reach into a root cause, explore further the criteria selected for the flag tests to have higher criteria rate, thus increase a little bit the imbalance, but not as much as up to 24% narrowing down truly suspicious amounts, but for the scope of this project, we believe our work suffice to answer our research question.

The first attempt to define a FNN model, yield bad results, but we managed to improve our information feed to the model, achieving the desirable accuracy and recall above 0.6, nevertheless, we might still have room to improve, either with the supervised model that we could re-do the model screening and hyperparameter tuning with the new target variable or with the FNN using keras tuner to perform hyperparameter tuning for neural networks.

The study evaluated various encoding methods and sampling techniques. Under sampling consistently bolstered recall but at the expense of lowered accuracy, while over sampling displayed higher accuracy but struggled to meet desired recall thresholds across all models assessed.

In summary, we improve our dataset understanding and through the insights gathered we leverage to refine our model, but we still have room to improve the model, using new forensic accounting tools, exploring more the dataset, testing new machine learning models or tuning our models.

6 REFERENCES

1. How Medicare and Medicaid fraud became a $100B problem for the U.S. (cnbc.com)
2. Healthcare Fraud Data Mining Methods: A Look Back and Look Ahead - PMC (nih.gov)
3. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9013219/
4. Medicare PART B dataset (cms.gov)
5. Order and referring dataset (cms.gov)
6. LEIE Downloadable Databases | Office of Inspector General | U.S. Department of Health and Human Services (hhs.gov)
7. All Exclusion Databases & More - Streamline Verify
8. Big Data fraud detection using multiple Medicare data sources | Journal of Big Data | Full Text (springeropen.com)
9. Identifying Physician Fraud in Healthcare with Open Data | SpringerLink
10. Nigrini, M. J. (2011). Forensic Analytics: Methods and Techniques for Forensic Accounting Investigations. Wiley.