Data Science Health Care Week 9

Group Member (Solo):

- *Name: Halit Ayberk Demir*
- *E-Mail: h.ayberk.demir.34@gmail.com*
- *University/Company: Middle East Technical University/Data Analyst at Mega Pharma*
- *Specialization: Data Science*

Problem Description

ABC is a pharmaceutical company and desires to understand the persistency of the drug per physician description. To solve this issue, ABC company reached an analytics company to automate this process of identification.

Data Understanding

Healthcare dataset is xlsx (Excel Workbook) which has 2 worksheets, at worksheet 1, 29 Features are explained individually. These features are divided into 6 buckets, this bucket and regarding features are:

1 - Target Variable

- Patient ID

2 - Demographics

- Age
- Race
- Region
- Ethnicity
- Gender
- IDN Indicator

3 - Provider Attributes

- NTM - Physician Specialty

4 - Clinical Factors

- NTM - T-Score
- Change in T Score

- NTM - Risk Segment
- Change in Risk Segment
- NTM - Multiple Risk Factors
- NTM - Dexa Scan Frequency
- NTM - Dexa Scan Recency
- Dexa During Therapy
- NTM - Fragility Fracture Recency
- Fragility Fracture During Therapy
- NTM - Glucocorticoid Recency
- Glucocorticoid Usage During Therapy

5 - Disease/Treatment Factor

- NTM - Injectable Experience
- NTM - Risk Factors
- NTM - Comorbidity
- NTM - Concomitancy
- Adherence

This features are in the columns of the dataset. Our actual dataset in worksheet number two. The actual dataset contains 3424 columns and 69 columns. When I investigated the dataset at first glance, I could see that most of the columns are either binary or categorical features.

```
RangeIndex: 3424 entries, 0 to 3423
Data columns (total 69 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Ptid                                        3424 non-null   object
 1   Persistency_Flag                            3424 non-null   object
 2   Gender                                      3424 non-null   object
 3   Race                                        3424 non-null   object
 4   Ethnicity                                   3424 non-null   object
 5   Region                                      3424 non-null   object
 6   Age_Bucket                                  3424 non-null   object
 7   Ntm_Speciality                              3424 non-null   object
 8   Ntm_Specialist_Flag                         3424 non-null   object
 9   Ntm_Speciality_Bucket                       3424 non-null   object
 10  Gluco_Record_Prior_Ntm                      3424 non-null   object
 11  Gluco_Record_During_Rx                      3424 non-null   object
 12  Dexa_Freq_During_Rx                         3424 non-null   int64
 13  Dexa_During_Rx                              3424 non-null   object
 14  Frag_Frac_Prior_Ntm                         3424 non-null   object
 15  Frag_Frac_During_Rx                         3424 non-null   object
 16  Risk_Segment_Prior_Ntm                      3424 non-null   object
 17  Tscore_Bucket_Prior_Ntm                     3424 non-null   object
 18  Risk_Segment_During_Rx                      3424 non-null   object
 19  Tscore_Bucket_During_Rx                     3424 non-null   object
...
 67  Risk_Recurring_Falls                        3424 non-null   object
 68  Count_Of_Risks                              3424 non-null   int64
dtypes: int64(2), object(67)
memory usage: 1.8+ MB
```

When I First read the dataset its memory usage is around 1.8 Mb, because pandas read the categorical values as objects. I wrote a function to turn the columns which has less than 50 unique values to categorical datatype:

```
df.loc[:, df.nunique() < 50] = df.loc[:, df.nunique() < 50].astype('category')
df.info()
```

```
 2   Gender                                              3424 non-null   category
 3   Race                                                3424 non-null   category
 4   Ethnicity                                           3424 non-null   category
 5   Region                                              3424 non-null   category
 6   Age_Bucket                                          3424 non-null   category
 7   Ntm_Speciality                                      3424 non-null   category
 8   Ntm_Specialist_Flag                                 3424 non-null   category
 9   Ntm_Speciality_Bucket                               3424 non-null   category
 10  Gluco_Record_Prior_Ntm                              3424 non-null   category
 11  Gluco_Record_During_Rx                              3424 non-null   category
 12  Dexa_Freq_During_Rx                                 3424 non-null   int64
 13  Dexa_During_Rx                                      3424 non-null   category
 14  Frag_Frac_Prior_Ntm                                 3424 non-null   category
 15  Frag_Frac_During_Rx                                 3424 non-null   category
 16  Risk_Segment_Prior_Ntm                              3424 non-null   category
 17  Tscore_Bucket_Prior_Ntm                             3424 non-null   category
 18  Risk_Segment_During_Rx                              3424 non-null   category
 19  Tscore_Bucket_During_Rx                             3424 non-null   category
...
 67  Risk_Recurring_Falls                                3424 non-null   category
 68  Count_Of_Risks                                      3424 non-null   category
dtypes: category(67), int64(1), object(1)
memory usage: 287.6+ KB
```

It's memory usage dropped to 287 Kb's as you can see. Memory usage dropped by 83.3 %. That proves, most of the columns in this data are either binary or consist of categorical data. In order to find the binary columns, I wrote a function:

```
# Binary Values

for name, column in df.items():
    if column.nunique() == 2:
        print(name)
```

And found out that 59 of the 69 columns are binary Data.

Same logic for finding the numeric values.

```
# Numeric Values

df.select_dtypes("int64").columns.to_list()
```

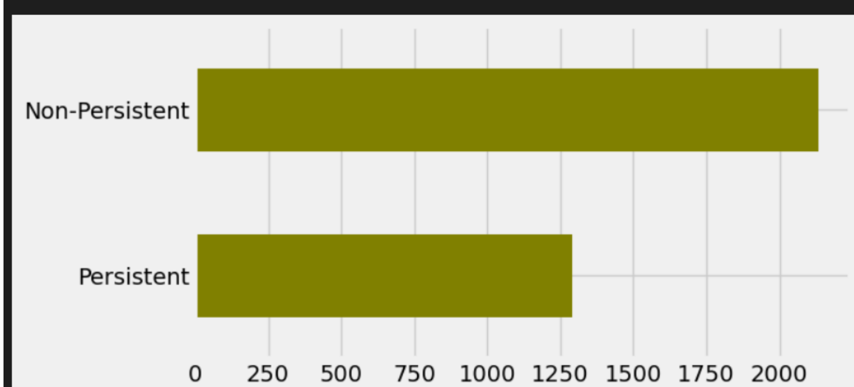
```
['Dexa_Freq_During_Rx', 'Count_Of_Risks']
```

It is clear that we have only two numerical data in our whole dataset.

Investigating Important Columns

Persistency Flag

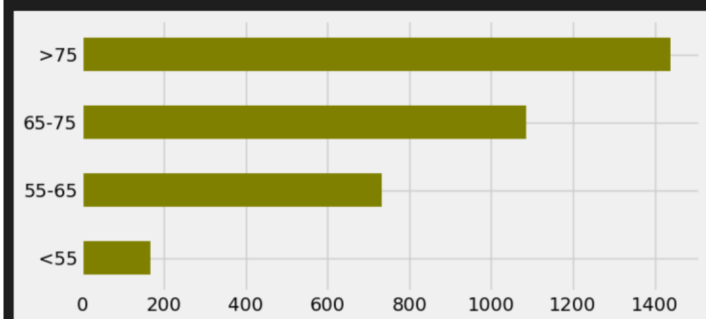
```
Value Counts for the Persistency_Flag
--------------------------------------------------
Non-Persistent    2135
Persistent        1289
Name: Persistency_Flag, dtype: int64
--------------------------------------------------
Column Desctiption for the Persistency_Flag
--------------------------------------------------
count               3424
unique                 2
top       Non-Persistent
freq                2135
Name: Persistency_Flag, dtype: object
--------------------------------------------------
```



Persistency flag is the target variable, we have 2135 Non persistent and 1289 persistent patients. It is clear that non persistent patients are more than persistent patients; thus, we have imbalanced target data.

Age

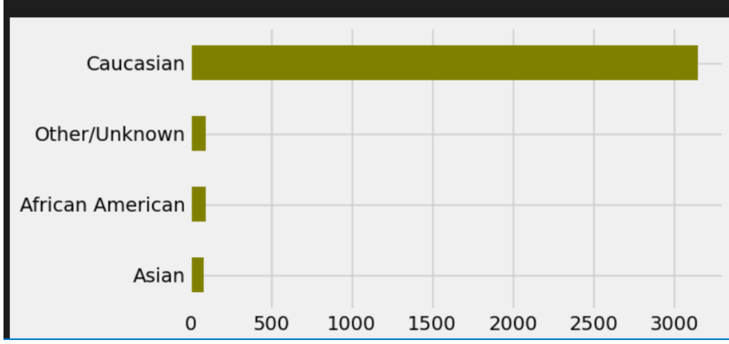
```
Value Counts for the Age_Bucket
--------------------------------------------------
>75       1439
65-75     1086
55-65      733
<55        166
Name: Age_Bucket, dtype: int64
--------------------------------------------------
Column Desctiption for the Age_Bucket
--------------------------------------------------
count     3424
unique       4
top        >75
freq      1439
Name: Age_Bucket, dtype: object
--------------------------------------------------
```



Patients over 75 years are dominating in this column and middle aged (<55) patients are the least common in this dataset.

Race

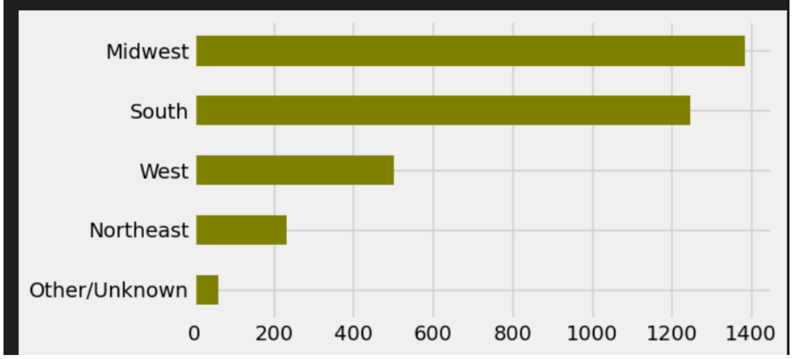
```
--------------------------------------------------
Caucasian           3148
Other/Unknown         97
African American      95
Asian                 84
Name: Race, dtype: int64
--------------------------------------------------
Column Desctiption for the Race
--------------------------------------------------
count          3424
unique            4
top       Caucasian
freq           3148
Name: Race, dtype: object
--------------------------------------------------
```



We have a lot of Caucasian patients. And notice that we have some ***unknown*** values in this column.

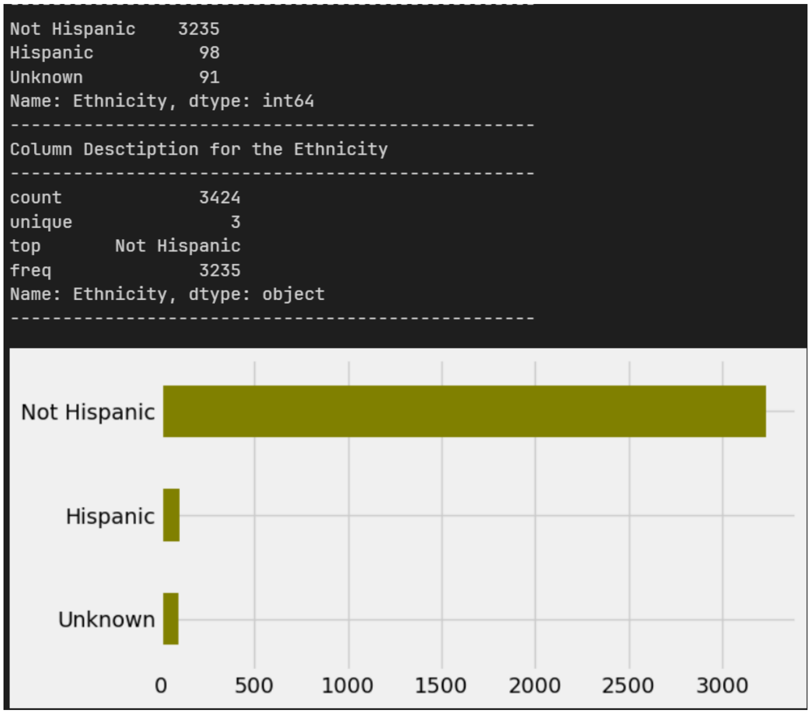
Region

```
Midwest          1383
South            1247
West              502
Northeast         232
Other/Unknown      60
Name: Region, dtype: int64
--------------------------------------------------
Column Desctiption for the Region
--------------------------------------------------
count        3424
unique          5
top       Midwest
freq         1383
Name: Region, dtype: object
--------------------------------------------------
```

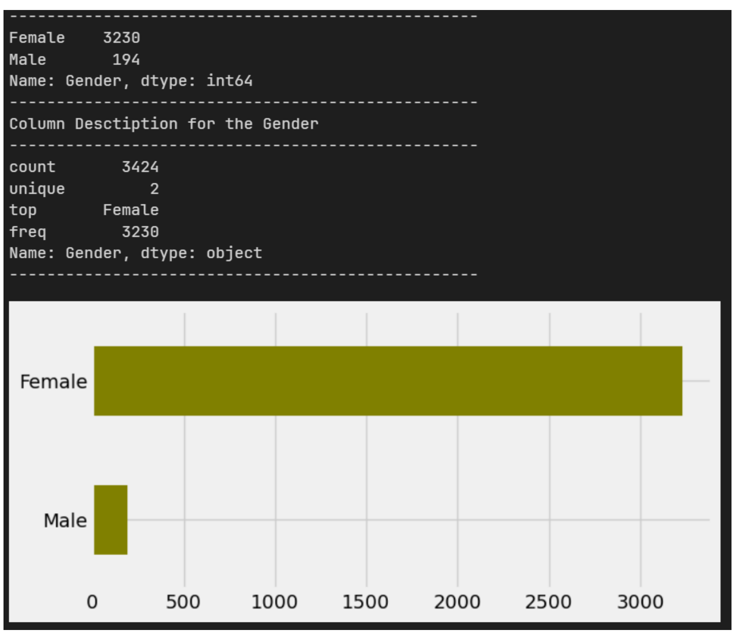


We have 60 unknown values!

Ethnicity



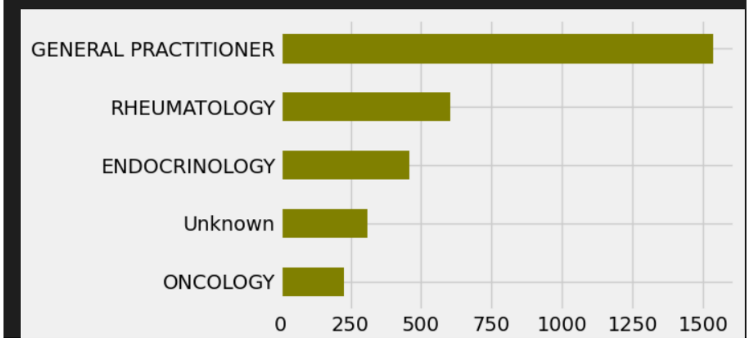
Gender



It is clear that this disease is much more common at females than males.

NTM Specialty

```
GENERAL PRACTITIONER    1535
RHEUMATOLOGY             604
ENDOCRINOLOGY            458
Unknown                  310
ONCOLOGY                 225
Name: Ntm_Speciality, dtype: int64
--------------------------------------------------
Column Desctiption for the Ntm_Speciality
--------------------------------------------------
count                     3424
unique                      36
top       GENERAL PRACTITIONER
freq                      1535
Name: Ntm_Speciality, dtype: object
--------------------------------------------------
```



NA Values

At first glance it seems like our dataset does not have NaN values, but we can see that there are some values categorized as Unknown and Other/Unkown values. This is a problem for our machine learning model and therefore we have to deal with these values. In order to check out the columns which has these values and the percentage of them:

```
def detect_none(x):
    if x in ["Unknown", "Other/Unknown"]:
        return None
    else:
        return x

df = df.applymap(detect_none)
```

```
none_columns = df.isna().sum()

percent_none = none_columns[none_columns > 0] / len(df) * 100

percent_none
```

```
Race                        2.832944
Ethnicity                   2.657710
Region                      1.752336
Ntm_Speciality              9.053738
Risk_Segment_During_Rx     43.720794
Tscore_Bucket_During_Rx    43.720794
Change_T_Score             43.720794
Change_Risk_Segment        65.099299
dtype: float64
```

Above you can see the percentages of NaN values. We have 8 columns that have NaN values, 4 of them have above 40% NaN values. For the columns which have more than 40 percent NaN values, we should get rid of those columns because any replacement of those rows will result in an error in our model. For the other NaN values one method could be filling them with the most frequent value in the desired column can be a solution. If the percentage is low enough, we can even delete those rows completely.

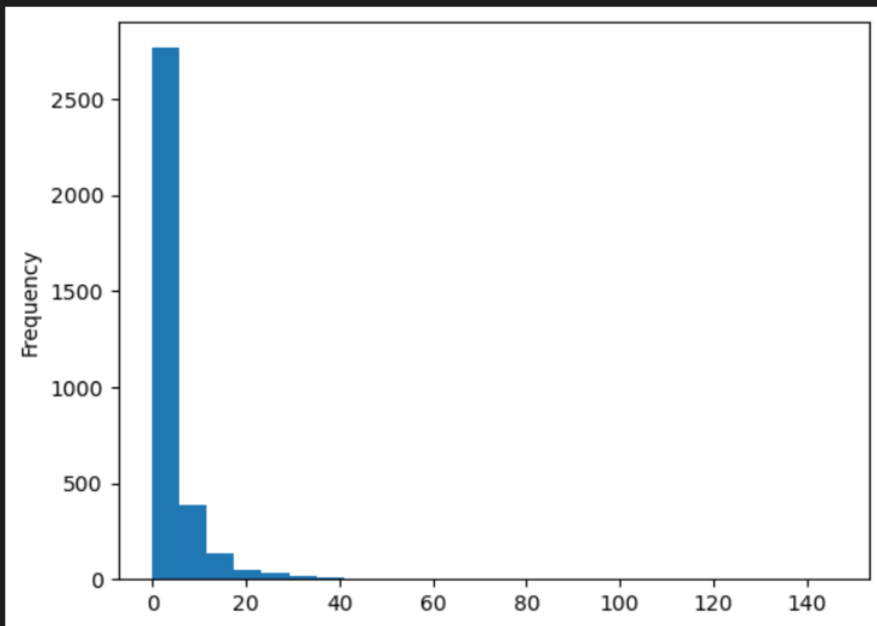
Numerical Values

As I mentioned above, we have 2 numeric columns, and their distributions are:

Dexa Frequency During Rx:

```
df["Dexa_Freq_During_Rx"].plot.hist(bins=25)
plt.show()
```

✓ 1.2s



It’s skewness and kurtosis are:

```
[13] df["Dexa_Freq_During_Rx"].skew()
```

✓ 0.0s

```
6.8087302112992285
```

```
[14] df["Dexa_Freq_During_Rx"].kurtosis()
```

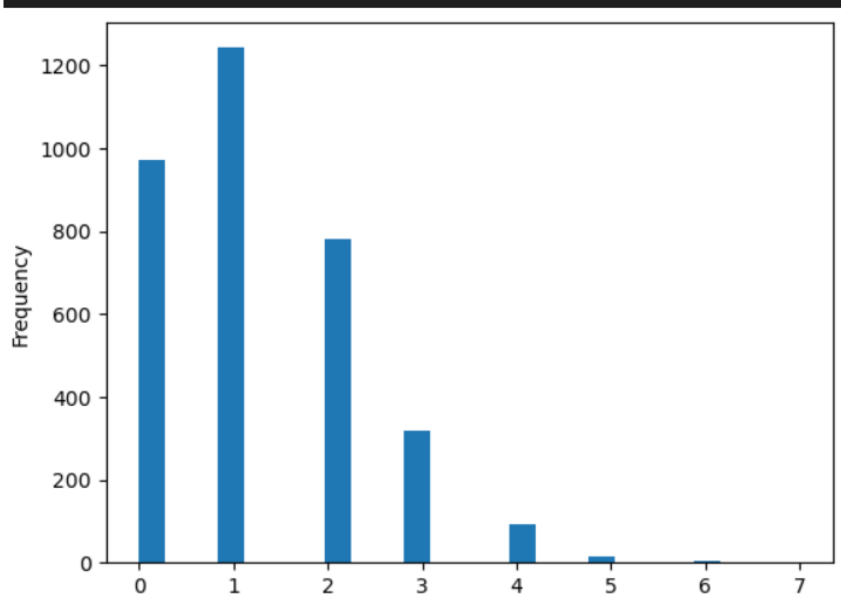
✓ 0.0s

```
74.75837754795428
```

Count Of Risks:

```
df["Count_Of_Risks"].plot.hist(bins=25)
plt.show()
```

✓ 0.1s



Its skewness and kurtosis are:

```
df["Count_Of_Risks"].skew()
```

✓ 0.0s

```
0.8797905232898707
```

```
df["Count_Of_Risks"].kurtosis()
```

✓ 0.0s

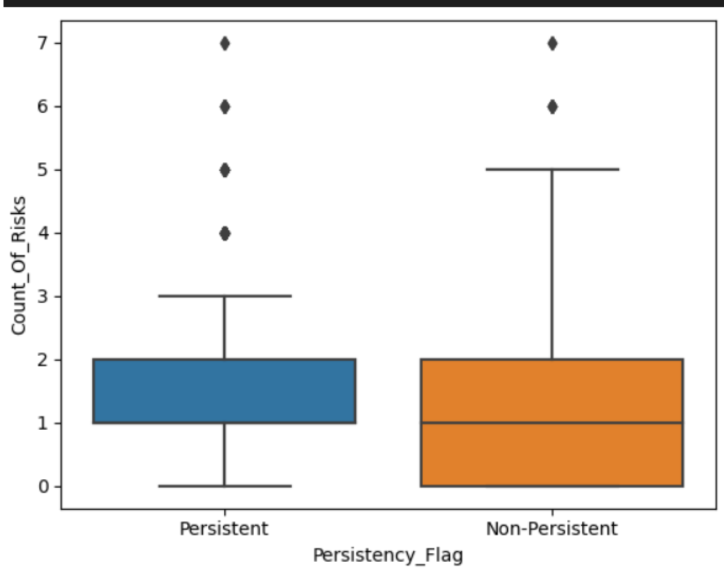
```
0.9004859968892842
```

Outliers

For outliers we can investigate the box plots.

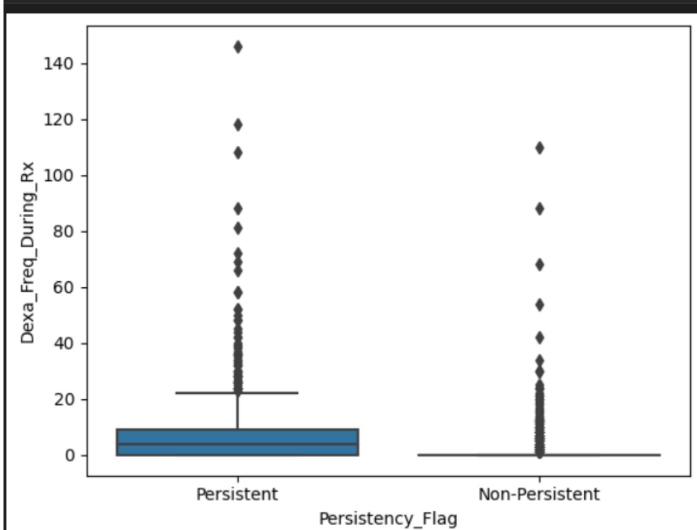
Count of Risks:

```
sns.boxplot(data=df, x="Persistency_Flag", y="Count_Of_Risks"
plt.show()
```



Dexa Frequency:

```
sns.boxplot(data=df, x="Persistency_Flag", y="Dexa_Freq_During_R
plt.show()
```



It is clear that we have outliers in our numerical data we can use interquartile ranges to find these outliers.

```
def find_outliers_IQR(df):

    q1=df.quantile(0.25)

    q3=df.quantile(0.75)

    IQR=q3-q1

    outliers = df[((df<(q1-2*IQR)) | (df>(q3+2*IQR)))]

    return outliers
```

```
find_outliers_IQR(df["Count_Of_Risks"]).count()
```

```
2
```

```
find_outliers_IQR(df["Dexa_Freq_During_Rx"]).count()
```

```
357
```

We have 2 outliers in the count of risks column and 357 outliers in the dexa frequency column. These outliers can be problematic for the machine learning model so getting rid of these outlier values can be the solution to this problem.

Cleaning The Data

I have fallowed two different approaches for cleaning the data and imputing the values.

1 – Cleaning NaN values and outliers.

NaN Values

```
none_columns = df.isna().sum()

percent_none = none_columns[none_columns > 0] / len(df) * 100

percent_none
```

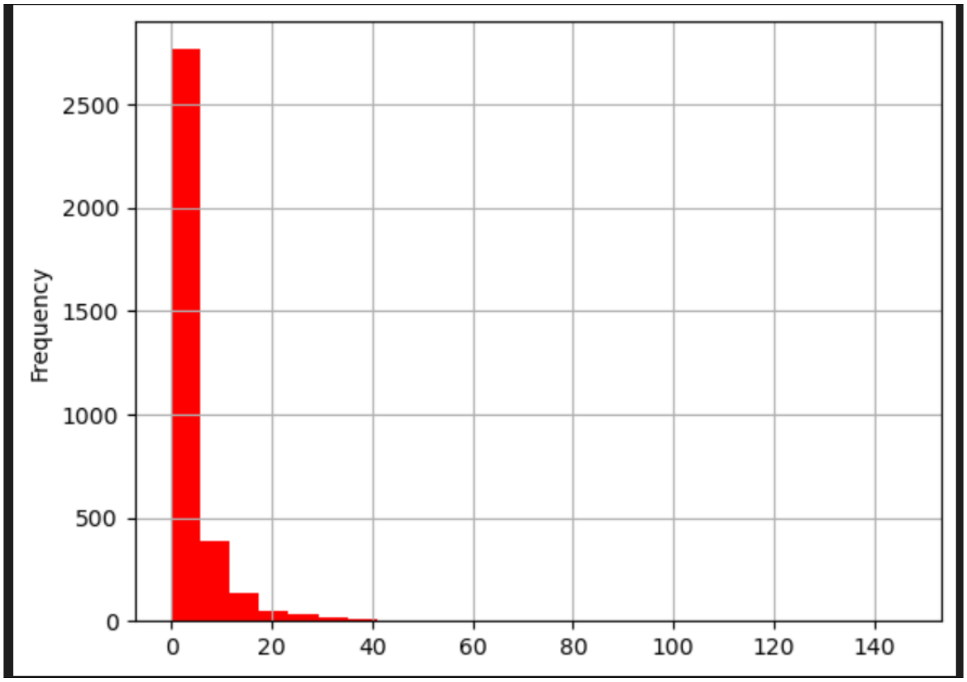
```
Race                       2.832944
Ethnicity                  2.657710
Region                     1.752336
Ntm_Speciality             9.053738
Risk_Segment_During_Rx    43.720794
Tscore_Bucket_During_Rx   43.720794
Change_T_Score            43.720794
Change_Risk_Segment       65.099299
dtype: float64
```

Here is the percentage of the NaN values in the columns since 4 columns has more than 40 percent of NaN values, I will drop these columns completely in order to not to affect the data completely. For the Race, Ethnicity, Region, Ntm Specialty column I will replace the values with the most frequent categorical values.

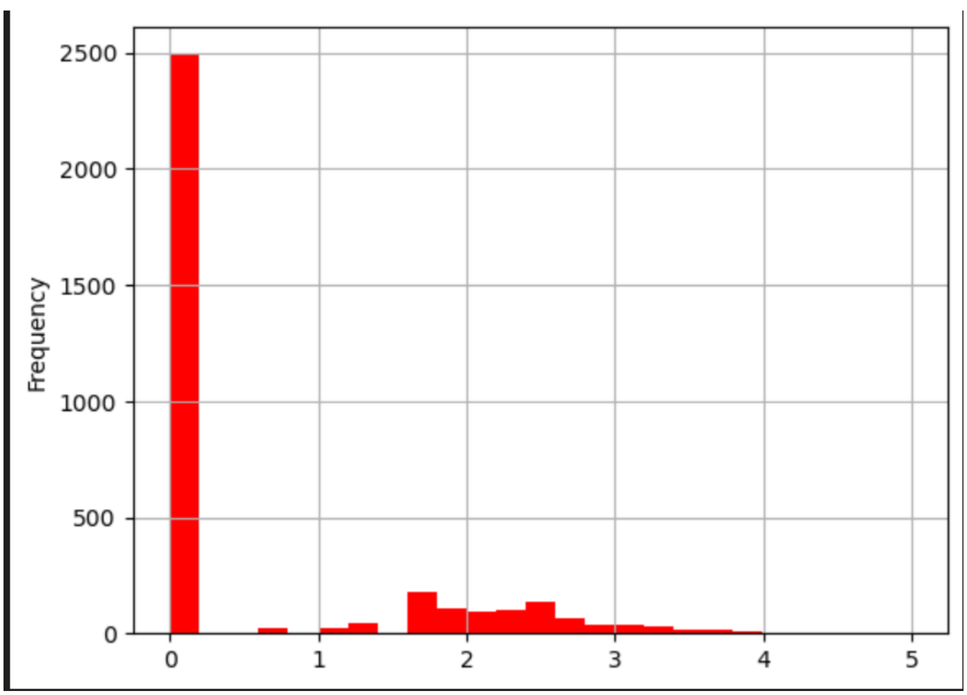
Outliers

Only outlier we have in the dataset is dexa freaquency during rx since it is not a normal distrubition, at first I used the log transformation to fit numeric data to normal distrubition.

Before Transformation



After Transformation



Then I used inquartile ranges to detect outliers and I dropped the outliers from the dataframe.

```
def find_outliers_IQR(df):
    """ Function for finding the outliers """
    q1=df.quantile(0.25)

    q3=df.quantile(0.75)

    IQR=q3-q1

    outliers = df[((df<(q1-2*IQR)) | (df>(q3+2*IQR)))]

    return outliers
```

For the next step I will convert all of the categorical data to numeric using labeling and one hot coding techniques.

2 – Weight of Evidence and Information Value.

Weight of Evidence and information value are values used to understand the predictive value of the independent features. It is used in binary classification techniques and we impute the categorical values with weight of evidence. And inspect the relative predictive power with information values. If sum of information values are less than 0.02, the feature is not relative. It is a powerful technique for feature selection but it is only used with logistic regression, so one of the model I will try will be logistic regression and I will use weight of evidence to impute values in logistic regression.

Here is an example of calculating woe/iv in one column;

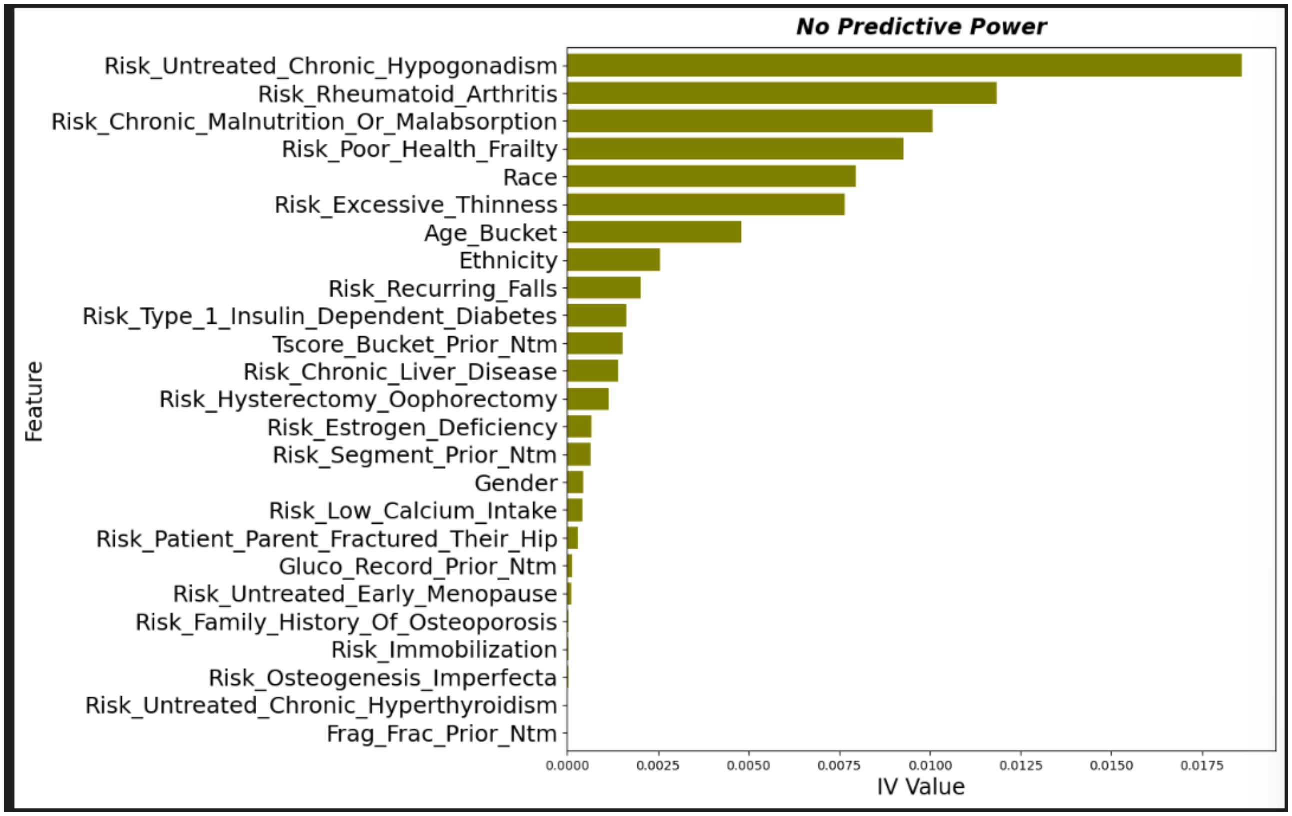
	Non-Persistent	Persistent	woe	iv
0	656	309	-0.2434	0.01618
1	770	467	0.009358	0.000032
2	467	313	0.109293	0.002765
3	179	138	0.249287	0.005925
4	50	41	0.310969	0.00266
5	6	9	0.914885	0.003853
6	3	3	0.509419	0.000476
7	1	1	0.509419	0.000159
Total	2132	1281	2.369231	0.03205

Here iv is 0.03 so the feature is a weak predictor.

In our dataset we have 35 columns which has an iv value greater than 0.02;

```
Region                                                             0.025565
Ntm_Speciality                                                     0.199183
Ntm_Specialist_Flag                                                0.082064
Ntm_Speciality_Bucket                                              0.136566
Gluco_Record_During_Rx                                             0.197153
Dexa_Freq_During_Rx                                                0.984861
Dexa_During_Rx                                                     0.754214
Frag_Frac_During_Rx                                                0.039780
Adherent_Flag                                                      0.063742
Idn_Indicator                                                      0.085094
Injectable_Experience_During_Rx                                    0.056867
Comorb_Encounter_For_Screening_For_Malignant_Neoplasms             0.372186
Comorb_Encounter_For_Immunization                                  0.370968
Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx    0.312116
Comorb_Vitamin_D_Deficiency                                        0.116784
Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified               0.202702
Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx 0.135976
Comorb_Long_Term_Current_Drug_Therapy                              0.518118
Comorb_Dorsalgia                                                   0.131785
Comorb_Personal_History_Of_Other_Diseases_And_Conditions           0.168811
Comorb_Other_Disorders_Of_Bone_Density_And_Structure               0.240664
Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias    0.107378
Comorb_Osteoporosis_without_current_pathological_fracture          0.082456
Comorb_Personal_history_of_malignant_neoplasm                      0.112425
Comorb_Gastro_esophageal_reflux_disease                            0.184019
...
Concom_Viral_Vaccines                                              0.209660
Risk_Smoking_Tobacco                                               0.060599
Risk_Vitamin_D_Insufficiency                                       0.025571
Count_Of_Risks                                                     0.031886
```

Graph of IV of the columns which has no predictive power.



Here is the dataset after imputing the WOE values.

	Ptid	Persistency_Flag	Region	Ntm_Speciality	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	Gluco_Record_During_Rx	Dexa_Freq_During_Rx	Dexa_D
0	P1	Persistent	0.072988	-0.234242	-0.248592	-0.239534	-0.280921	-0.470083	-
1	P2	Non-Persistent	0.072988	-0.234242	-0.248592	-0.239534	-0.280921	-0.470083	-
2	P3	Non-Persistent	-0.188352	-0.234242	-0.248592	-0.239534	-0.280921	-0.470083	-
3	P4	Non-Persistent	-0.188352	-0.234242	-0.248592	-0.239534	0.713455	-0.470083	-
4	P5	Non-Persistent	-0.188352	-0.234242	-0.248592	-0.239534	0.713455	-0.470083	-

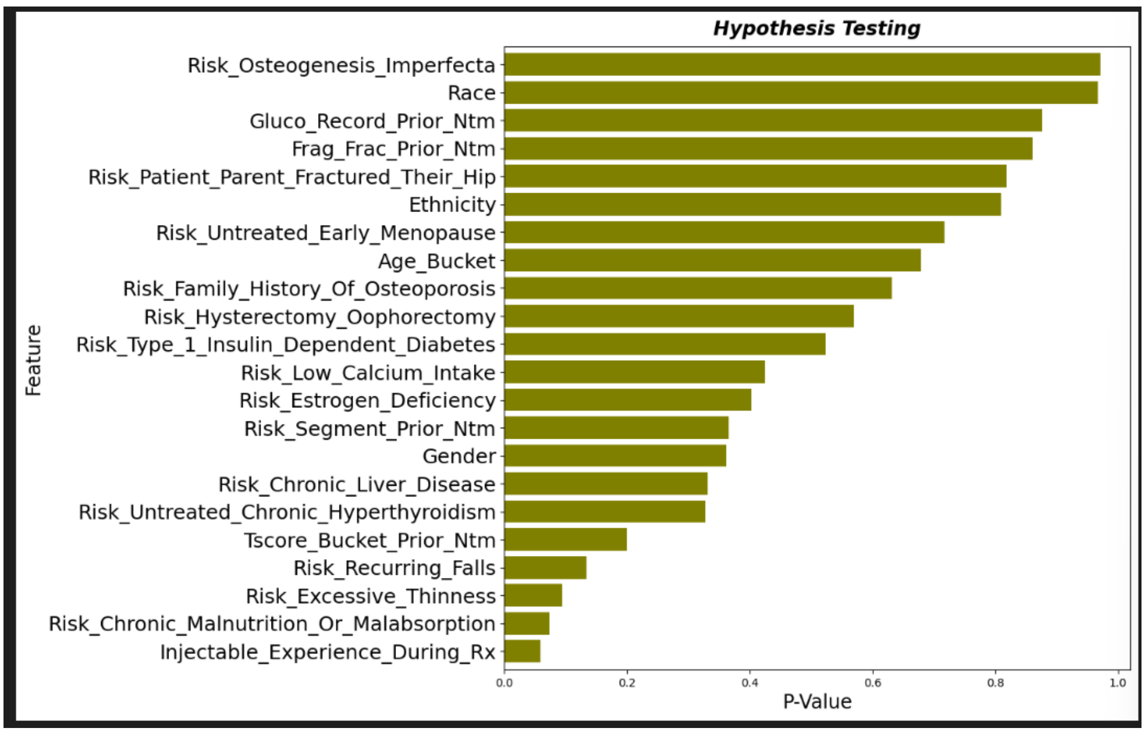
3– Hypothesis Testing for Feature Selection.

Since most of the features are categorical, we can use chi square test for investigating association between the target and the features.

- H0: There is no association between target and the feature.
- H1: There is an association between target and the feature.

If p is below 0.05, we reject the null hypothesis

Here are the p values of the columns who failed to reject null hypothesis:

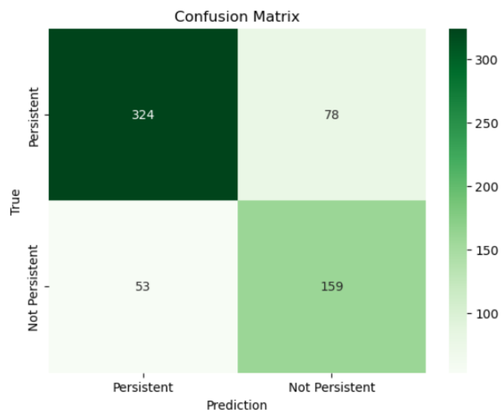


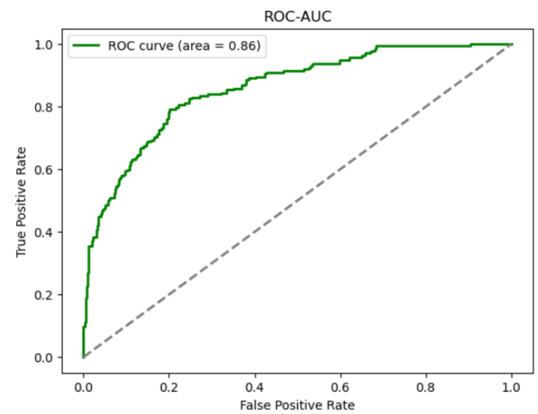
4– Model Selection.

In this dataset I will use classification algorithms, and I will use WOE in logistic regression.

Here are the models that is trained and related metrics;

1 – Model 1 Logistic Regression. (Base Model)

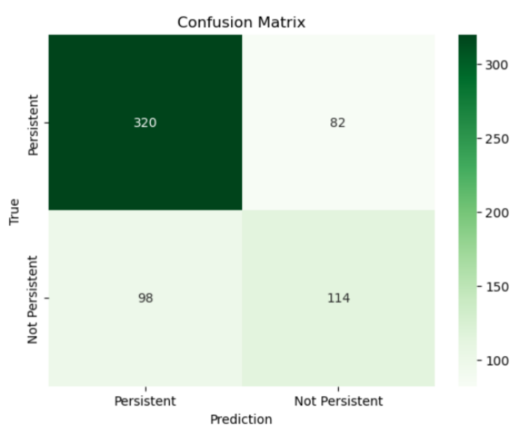


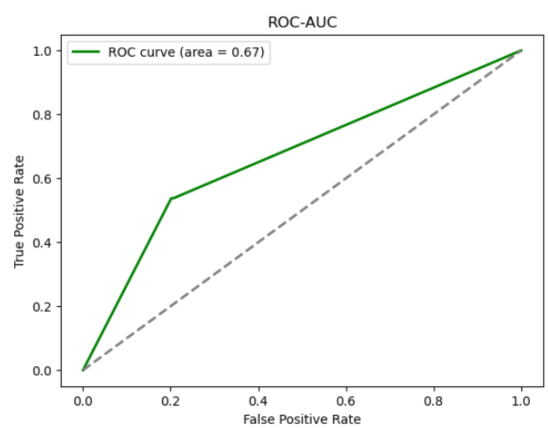


Metrics are;

```
Accuracy:  0.7866449511400652
Precision:  0.6708860759493671
Recall:  0.75
F1 Score:  0.7082405345211581
AUC-ROC:  0.7779850746268656
```

2 – Model 2 Decision Tree.



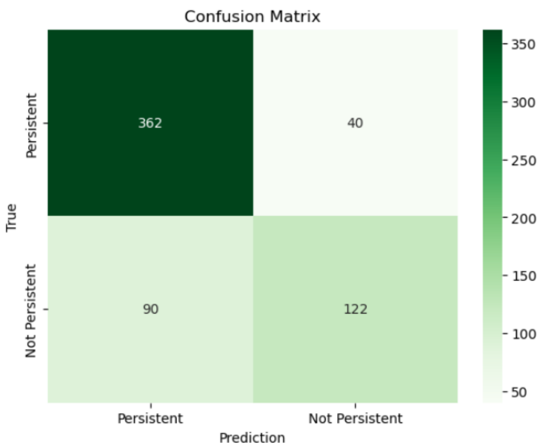


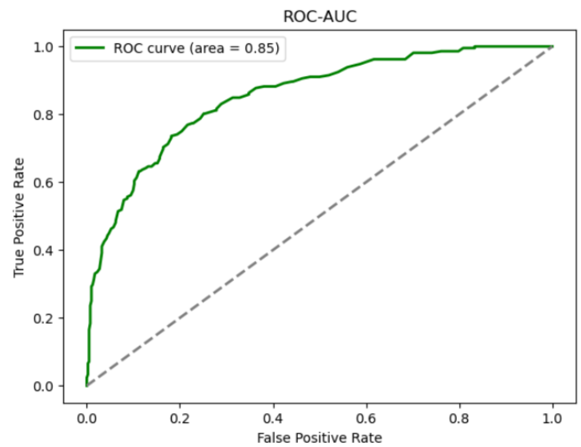
Metrics are;

```
Accuracy:  0.7068403908794788
Precision:  0.5816326530612245
```

```
Recall:  0.5377358490566038
F1 Score:  0.5588235294117646
AUC-ROC:  0.6668778747770582
```

3 – Model 3 Random Forest Classifier.

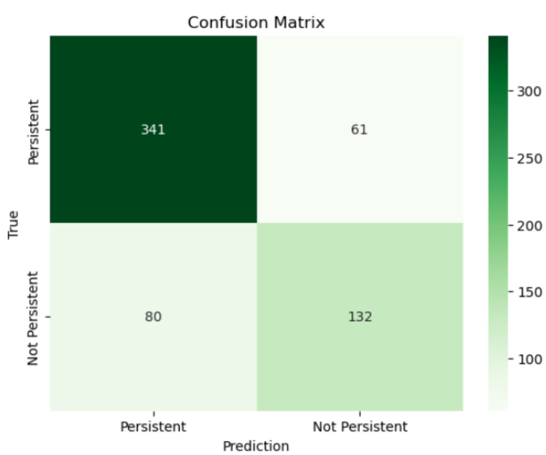


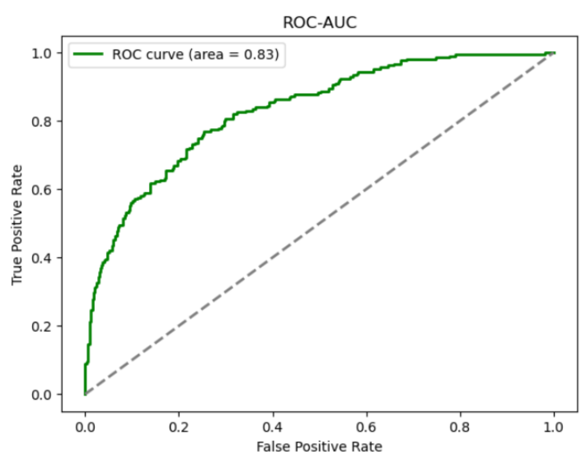


Metrics are;

```
Accuracy:  0.7882736156351792
Precision:  0.7530864197530864
Recall:  0.5754716981132075
F1 Score:  0.6524064171122994
AUC-ROC:  0.7379846052755092
```

4 – Model 4 Gradient Boosting.

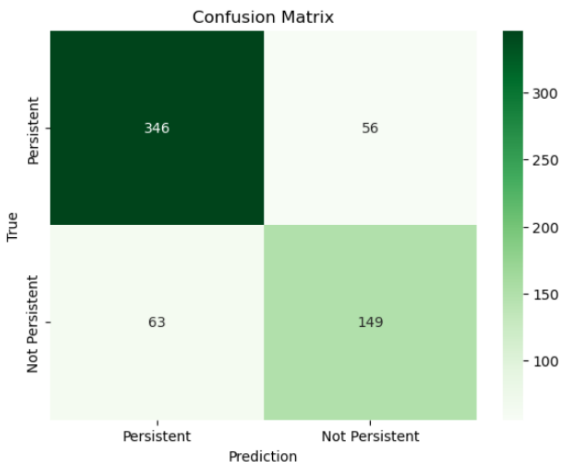


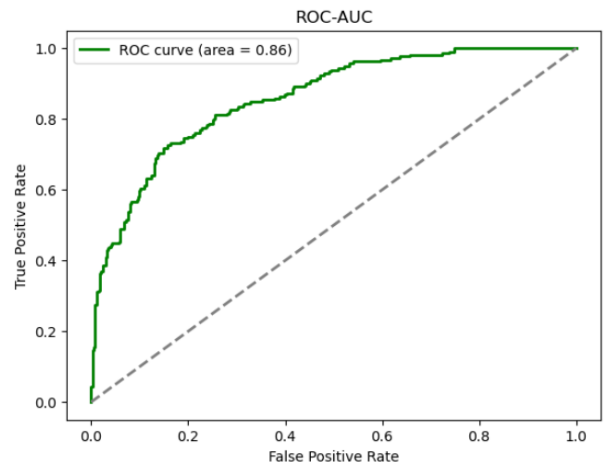


Metrics are;

```
Accuracy:  0.7703583061889251
Precision:  0.6839378238341969
Recall:  0.6226415094339622
F1 Score:  0.6518518518518519
AUC-ROC:  0.735450107950812
```

5 – Model 5 Support Vector Machines.

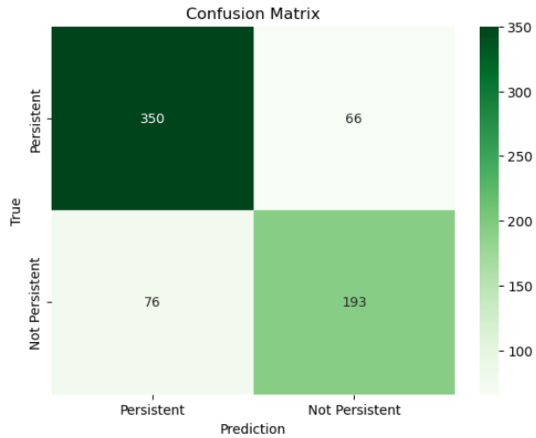


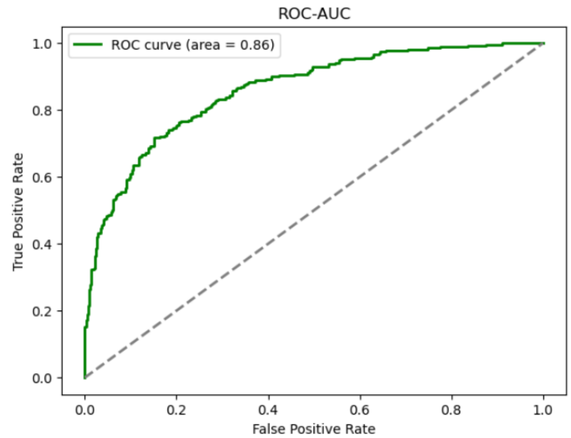


Metrics are;

```
Accuracy:  0.8061889250814332
Precision:  0.7268292682926829
Recall:  0.7028301886792453
F1 Score:  0.7146282973621103
AUC-ROC:  0.7817633530460902
```

6 – Model 6 Logistic Regression with WOE imputed Values.

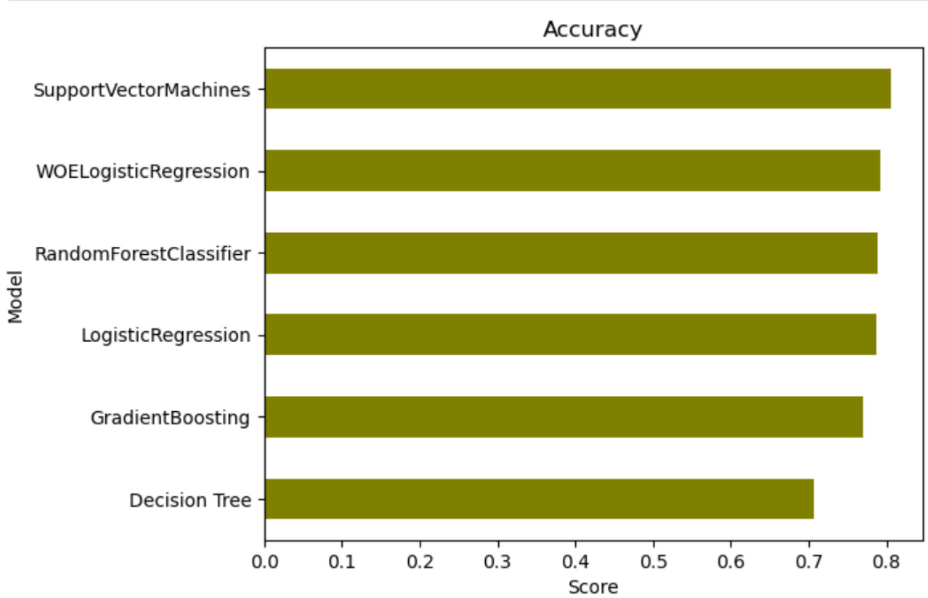


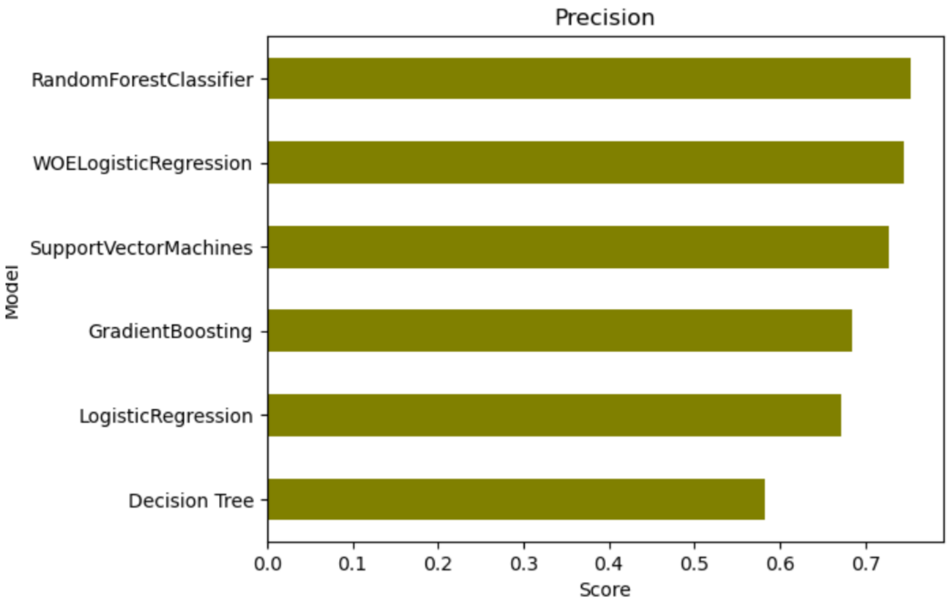


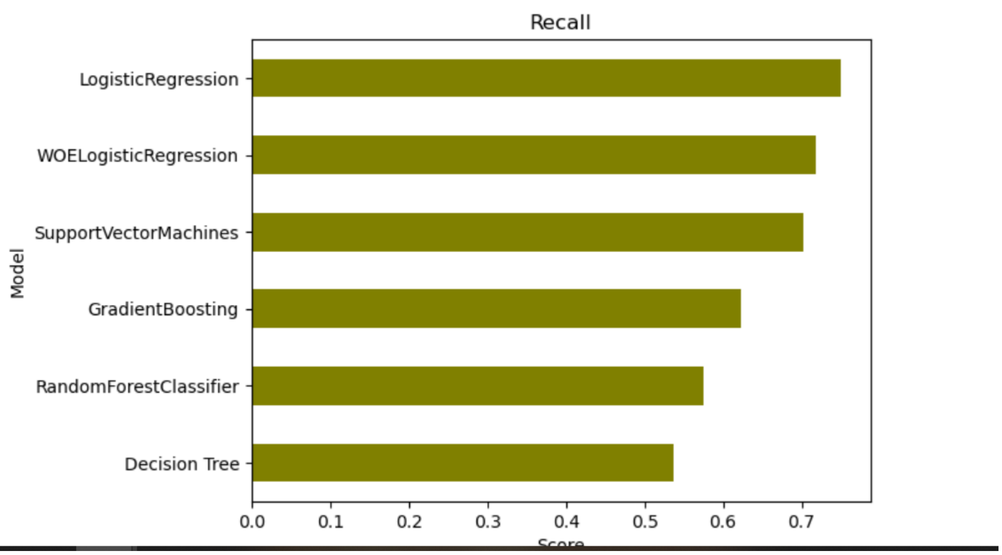
Metrics are;

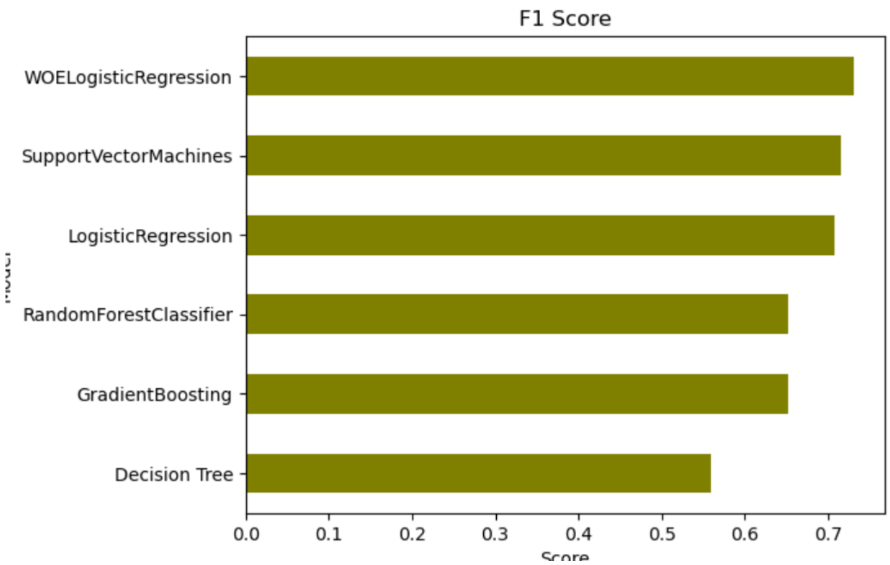
```
Accuracy:  0.7927007299270074
Precision:  0.7451737451737451
Recall:  0.7174721189591078
F1 Score:  0.731060606060606
AUC-ROC:  0.7794091364026308
```

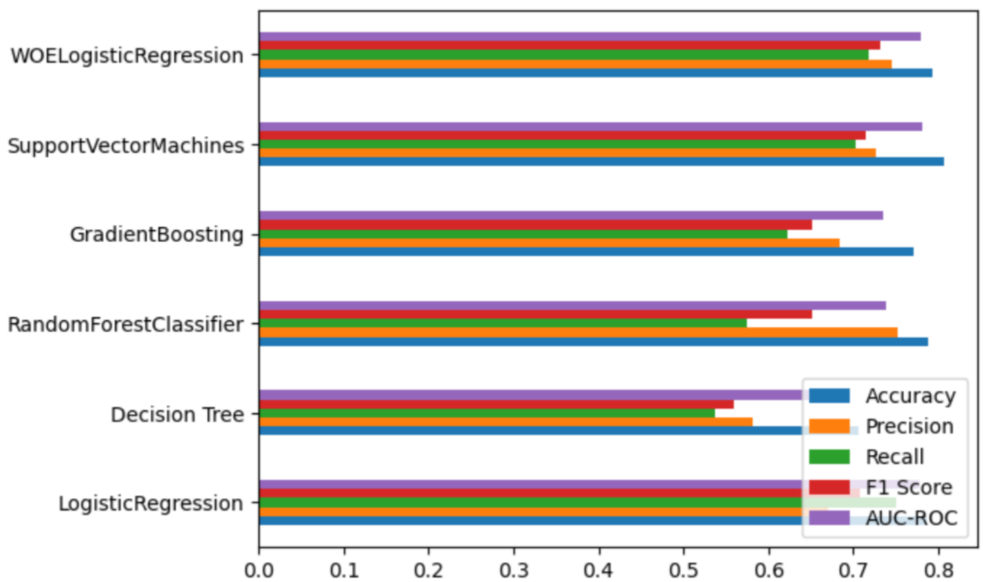
Here are the models that I provided, on the first glance woe imputed logistic regression and support vector machines has significantly more F1 values with 0,71 support vector machines and 0,73 with logistic regression. These both have accuracy around 80%. Let's compare them in detail with graphs.




Precision
RandomForestClassifier
WOELogisticRegression
SupportVectorMachines
GradientBoosting
LogisticRegression
Decision Tree
Model
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Score


Recall
LogisticRegression
WOELogisticRegression
SupportVectorMachines
GradientBoosting
RandomForestClassifier
Decision Tree
Model
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7





These metrics show that woe imputed logistic regression dominates other models. But SVM also has

A significant F1 score and it is higher in accuracy by 0,1 and has an easier implementation. Hence with its ease in implementation and high value metrics, SVM is the smartest choise between the models.

Thank you!

Halit Ayberk DEMIR

Github Repo: https://github.com/demirayberk/Data-Science-HealthCare