**NATIONWIDE INSURANCE – TECHNICAL ASSESSMENT**

**Aytekin Yıldızhan**

**1. INTRODUCTION**

In this technical assessment, we analzye the restaurant inspections that took place in the Las Vegas metropolitan area. We have TRAIN\_SET\_2021.csv and TEST\_SET\_2021.csv and our aim is to explore the possibility of building a minimally viable product (MVP) model to predict the outcome of a restaurant’s next inspection.

**2. TASKS**

**2.1. DATA ANALYSIS and VISULATION**

There are 15673 records in TRAIN\_SET\_2021 and 7505 records in TEST\_SET\_2021. We have 28 columns. These are the columns:

Column Non-Null Count Dtype

--- ------ -------------- -----

0 RESTAURANT\_SERIAL\_NUMBER 15673 non-null object

1 RESTAURANT\_PERMIT\_NUMBER 15673 non-null object

2 RESTAURANT\_NAME 15608 non-null object

3 RESTAURANT\_LOCATION 15473 non-null object

4 RESTAURANT\_CATEGORY 15543 non-null object

5 ADDRESS 15603 non-null object

6 CITY 15437 non-null object

7 STATE 15464 non-null object

8 ZIP 15614 non-null object

9 CURRENT\_DEMERITS 15457 non-null float64

10 CURRENT\_GRADE 15365 non-null object

11 EMPLOYEE\_COUNT 15580 non-null float64

12 MEDIAN\_EMPLOYEE\_AGE 15639 non-null float64

13 MEDIAN\_EMPLOYEE\_TENURE 15376 non-null float64

14 INSPECTION\_TIME 15490 non-null object

15 INSPECTION\_TYPE 15452 non-null object

16 INSPECTION\_DEMERITS 15419 non-null object

17 VIOLATIONS\_RAW 15508 non-null object

18 RECORD\_UPDATED 15554 non-null object

19 LAT\_LONG\_RAW 15658 non-null object

20 FIRST\_VIOLATION 15461 non-null float64

21 SECOND\_VIOLATION 15588 non-null float64

22 THIRD\_VIOLATION 15612 non-null float64

23 FIRST\_VIOLATION\_TYPE 15527 non-null object

24 SECOND\_VIOLATION\_TYPE 15406 non-null object

25 THIRD\_VIOLATION\_TYPE 15500 non-null object

26 NUMBER\_OF\_VIOLATIONS 15504 non-null object

27 NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW 15633 non-null object

print(len(train.RESTAURANT\_SERIAL\_NUMBER.unique())) = 15673

As you can see, RESTAURANT\_SERIAL\_NUMBER column is the unique. Let us start with the ouliers and null values. Here is the nul values:

train.isnull().sum().sort\_values(ascending=False)

train.isnull().sum().sum()

CURRENT\_GRADE 308

MEDIAN\_EMPLOYEE\_TENURE 297

SECOND\_VIOLATION\_TYPE 267

INSPECTION\_DEMERITS 254

CITY 236

INSPECTION\_TYPE 221

CURRENT\_DEMERITS 216

FIRST\_VIOLATION 212

STATE 209

RESTAURANT\_LOCATION 200

INSPECTION\_TIME 183

THIRD\_VIOLATION\_TYPE 173

NUMBER\_OF\_VIOLATIONS 169

VIOLATIONS\_RAW 165

FIRST\_VIOLATION\_TYPE 146

RESTAURANT\_CATEGORY 130

RECORD\_UPDATED 119

EMPLOYEE\_COUNT 93

SECOND\_VIOLATION 85

ADDRESS 70

RESTAURANT\_NAME 65

THIRD\_VIOLATION 61

ZIP 59

NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW 40

MEDIAN\_EMPLOYEE\_AGE 34

LAT\_LONG\_RAW 15

RESTAURANT\_PERMIT\_NUMBER 0

RESTAURANT\_SERIAL\_NUMBER 0

4027

We have 4027 null values. We have to drop this rows. Then we check the outliers. Figure 1 shows the distribution of null values.

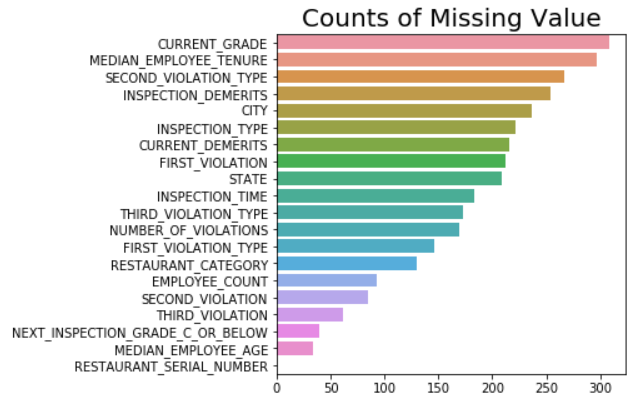


Figure 1. Missing Values

train.NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW.value\_counts()

0 13143

1 2484

-3 1

Goat 1

7 1

9 1

4 1

3 1

Name: NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW, dtype: int64

As you can see, -3, Goat, 7, 9, 4 and 3 are outliers.

train**.**CURRENT\_GRADE**.**value\_counts()

A 14915

B 215

C 104

X 75

O 32

N 13

7 2

K 1

A+ 1

U 1

I 1

EIEIO 1

UPN 1

VPN 1

NASA 1

.\<><1@#&| 1

Name: CURRENT\_GRADE, dtype: int64

As you can see above, other than A, B, C, X, O and N, these are outliers.

train**.**FIRST\_VIOLATION\_TYPE**.**value\_counts()

Critical 7194

Major 6735

Non-Major 1588

Imminent Health Hazard 3

Extra Crispy 1

Major-ish 1

To Infinity and Beyond 1

Not Sure 1

Radical 1

Excellent 1

Bullwinkle 1

Name: FIRST\_VIOLATION\_TYPE, dtype: int64

As you can see, we just consider the critical, major, non major and Imminent Health Hazard.

After that, we drop some irrelevant columns for building MVP.

train**.**drop('RESTAURANT\_PERMIT\_NUMBER',axis**=**1,inplace**=True**)

train**.**drop('VIOLATIONS\_RAW',axis**=**1,inplace**=True**)

train**.**drop('RESTAURANT\_NAME',axis**=**1,inplace**=True**)

train**.**drop('RESTAURANT\_LOCATION',axis**=**1,inplace**=True**)

train**.**drop('ADDRESS',axis**=**1,inplace**=True**)

train**.**drop('ZIP',axis**=**1,inplace**=True**)

train**.**drop('RECORD\_UPDATED',axis**=**1,inplace**=True**)

train**.**drop('LAT\_LONG\_RAW',axis**=**1,inplace**=True**)

Then we have 20 columns left. Then we identify the columns. Analyzing the train\_set, there   
are 17 features that can affect the prediction results.

identifier\_feature **=** ['RESTAURANT\_SERIAL\_NUMBER']

continuous\_features **=** ['MEDIAN\_EMPLOYEE\_AGE', 'MEDIAN\_EMPLOYEE\_TENURE']

nominal\_features **=** ['RESTAURANT\_CATEGORY', 'CITY', 'STATE','CURRENT\_GRADE','INSPECTION\_TYPE','FIRST\_VIOLATION', 'SECOND\_VIOLATION','THIRD\_VIOLATION', 'FIRST\_VIOLATION\_TYPE', 'SECOND\_VIOLATION\_TYPE', 'THIRD\_VIOLATION\_TYPE']

numeric\_feactures **=** ['CURRENT\_DEMERITS', 'EMPLOYEE\_COUNT','INSPECTION\_DEMERITS','NUMBER\_OF\_VIOLATIONS']

target **=** ['NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW']

selected\_features **=** nominal\_features **+** numeric\_feactures **+** continuous\_features **+** target

Figure 2 shows that the correlation of train data.

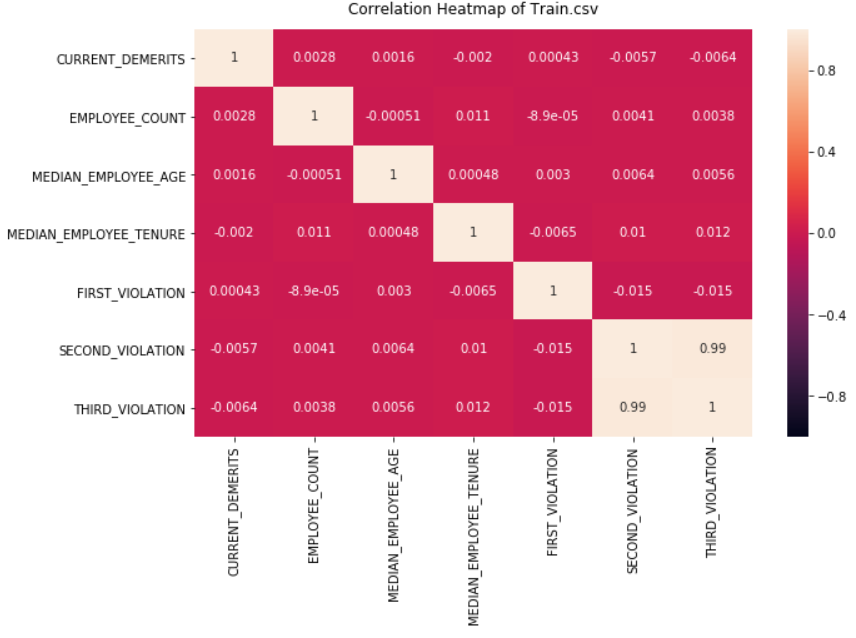


Figure 2. Correlation Map

Next, we combine train and test set because all data must enter the preprocessing section. We drop the null values and we make an assumptions.

df\_no\_null **=** df**.**dropna() # we drop the null values.

**2.2. ASSUMPTIONS**

Our assumptions are below.

1) df\_no\_null **=** df\_no\_null[df\_no\_null['INSPECTION\_TYPE']**.**isin(["Routine Inspection", "Re-inspection"])]

*# "Routine Inspection", "Re-inspection"*

2) df\_no\_null **=** df\_no\_null[df\_no\_null['CURRENT\_GRADE']**.**isin(["A", "B", "C", "X", "O", "N"])]

*# "A", "B", "C", "X", "O", "N"*

3) df\_no\_null **=** df\_no\_null[df\_no\_null['NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW']**.**isin(["0", "1"])]

# 0 and 1

We filter out the negative values.

4) df\_no\_null **=** df\_no\_null[(0 **<** df\_no\_null['FIRST\_VIOLATION']) **&** (df\_no\_null['FIRST\_VIOLATION'] **<** 311)]

df\_no\_null **=** df\_no\_null[(0 **<** df\_no\_null['SECOND\_VIOLATION']) **&** (df\_no\_null['SECOND\_VIOLATION'] **<** 311)]

df\_no\_null **=** df\_no\_null[(0 **<** df\_no\_null['THIRD\_VIOLATION']) **&** (df\_no\_null['THIRD\_VIOLATION'] **<** 311)]

df\_no\_null **=** df\_no\_null[(0 **<=** df\_no\_null['CURRENT\_DEMERITS']) **&** (df\_no\_null['CURRENT\_DEMERITS'] **<** 200)]

df\_no\_null **=** df\_no\_null[(0 **<** df\_no\_null['EMPLOYEE\_COUNT']) **&** (df\_no\_null['EMPLOYEE\_COUNT'] **<** 100)]

df\_no\_null**.**STATE**.**value\_counts() # we select ‘Nevada’

Nevada 20237

SK 5

Star Trek 1

Washington 1

New Mexico 1

Montana 1

TT 1

NeVaDa 1

Nevada? 1

Name: STATE, dtype: int64

5) df\_no\_null **=** df\_no\_null[df\_no\_null['STATE']**==**'Nevada']

**2.3. DATA PREPROCESSING**

In this part, we choose MinMaxScaler() for preprocessing.

**from** sklearn **import** preprocessing

df\_disc **=** pd**.**DataFrame()

*# Discretization*

**for** i **in** continuous\_features:

*# continuous\_features = ['MEDIAN\_EMPLOYEE\_AGE', 'MEDIAN\_EMPLOYEE\_TENURE']*

disc **=** pd**.**cut(df\_no\_null[i], bins**=**10, labels**=**np**.**arange(10),right**=False**)

df\_disc **=** pd**.**concat([df\_disc, disc], axis**=**1)

*# Concatenate numeric features and discretized features*

**for** i **in** numeric\_feactures:

*#numeric\_feactures = ['CURRENT\_DEMERITS', 'EMPLOYEE\_COUNT', 'INSPECTION\_DEMERITS', 'NUMBER\_OF\_VIOLATIONS']*

df\_disc **=** pd**.**concat([df\_disc, df\_no\_null[i]], axis**=**1)

*# Normalization*

*#x = df\_disc.values #returns a numpy array*

*#min\_max\_scaler = MinMaxScaler()*

*#x\_scaled = min\_max\_scaler.fit\_transform(x)*

x **=** df\_disc**.**values *#returns a numpy array*

min\_max\_scaler **=** preprocessing**.**MinMaxScaler()

x\_scaled **=** min\_max\_scaler**.**fit\_transform(x)

Then we binarize the nominal features. We use get\_dummies function.

""" nominal\_features = ['RESTAURANT\_CATEGORY', 'CITY', 'STATE', 'CURRENT\_GRADE','INSPECTION\_TYPE','FIRST\_VIOLATION', 'SECOND\_VIOLATION', 'THIRD\_VIOLATION','FIRST\_VIOLATION\_TYPE','SECOND\_VIOLATION\_TYPE','THIRD\_VIOLATION\_TYPE']

"""

restaurant\_category **=** pd**.**get\_dummies(df\_no\_null["RESTAURANT\_CATEGORY"], drop\_first**=True**)

city **=** pd**.**get\_dummies(df\_no\_null["CITY"], drop\_first**=True**)

state **=** pd**.**get\_dummies(df\_no\_null["STATE"], drop\_first**=True**)

current\_grade **=** pd**.**get\_dummies(df\_no\_null["CURRENT\_GRADE"], drop\_first**=True**)

inspection\_type **=** pd**.**get\_dummies(df\_no\_null["INSPECTION\_TYPE"], drop\_first**=True**)

first\_violation **=** pd**.**get\_dummies(df\_no\_null["FIRST\_VIOLATION"], drop\_first**=True**)

second\_violation **=** pd**.**get\_dummies(df\_no\_null["SECOND\_VIOLATION"], drop\_first**=True**)

third\_violation **=** pd**.**get\_dummies(df\_no\_null["THIRD\_VIOLATION"], drop\_first**=True**)

first\_violation\_type **=** pd**.**get\_dummies(df\_no\_null["FIRST\_VIOLATION\_TYPE"], drop\_first**=True**)

second\_violation\_type **=** pd**.**get\_dummies(df\_no\_null["SECOND\_VIOLATION\_TYPE"], drop\_first**=True**)

third\_violation\_type **=** pd**.**get\_dummies(df\_no\_null["THIRD\_VIOLATION\_TYPE"], drop\_first**=True**)

Next, we concatenate these new columns to our original train data then we drop nominal\_features columns. Finally we drop RESTAURANT\_SERIAL\_NUMBER column since its type is object.

Our data is ready to train.

**3. ATTEMPTING TO BUILD AN MVP MODEL**

Our data consists of train.csv and test.csv. First of all we split and we use train.csv to train.

df2\_test **=** df2[df2['ds\_type']**==**'Test']

df2\_train **=** df2[df2['ds\_type']**==**'Train']

*# drop the ds\_type*

df2\_train **=** df2\_train**.**drop(['ds\_type'], axis**=**1)

df2\_test **=** df2\_test**.**drop(['ds\_type'], axis**=**1)

We use 12 classifier machine learning techniques. We consider multiple performance evaluation metrics such as precision, recall, f1-score, and coefficient matrix. Here is our accuracy results. These detailed results are in Nationwide.html file.

**Classifier Accuracy**

1 SVM 0.842270

4 Logistic Regression 0.842270

3 Random Forest 0.840439

5 Gradient Boosting 0.839916

6 Ada Boost 0.839916

8 Linear Discriminant Analysis 0.835470

11 SGD 0.759090

0 Knn 0.741564

10 MLP 0.726654

2 Decision Tree 0.721161

7 Gaussian NB 0.204551

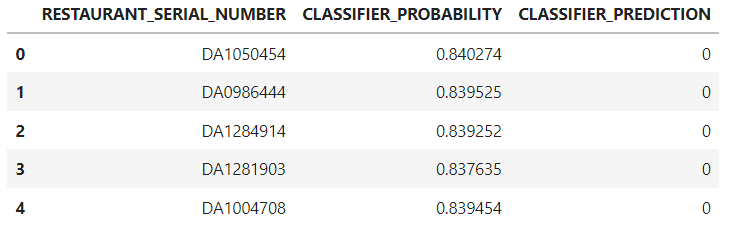
9 Quadratic Discriminant Analysis 0.169500

As you can see, SVM and Logistic Regression is the best accuracy. We choose SVM models to test test.csv.

In this assessment, my recommendation is that because of high number of outliers, none of classifiers cannot be reliable to be used in an MVP. The main reason is that the dataset is imbalance such as NaN (null), zero, negative or different values are inside the data. Therefore almost there are 20% missing value in this dataset.

**4.) Apply Our Model to The Test Set**

We select SVM and our result like this.



Then we save this result.

results**.**to\_csv('predictions\_Yildizhan\_Aytekin.csv', sep **=** ',',index **=** **False**) *#*

**5. Recommendations**

Based on the dataset, in my opinion MVP model is not recommended. Almost %20 of data is discarded by some reason. In test set, all of SVM classifier prediction is zero. If we have more time, we try to different classifier techniques.

To make the dataset more consistent, one of the way is to collect more data which has both zero and one. (NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW). Also according to correlation map, we add the features FIRST\_VIOLATION, SECOND\_VIOLATION, and THIRD\_VIOLATION. These features are more effectively to NEXT\_INSPECTION\_GRADE\_C\_OR\_BELOW. Also we do not use VIOLATIONS\_RAW since there is no information about violation numbers. Having more features like these features can improve the accuracy of the model.