Exploratory Data Analysis

```
# Libraries Imported

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


%matplotlib inline
plt.style.use('bmh')

# To display multiple outputs in a cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
from sklearn.preprocessing import StandardScaler

print("Libraries Imported")
```

```
Libraries Imported
```

Double-click (or enter) to edit

```
# Importing Dataset in the Environment
data = pd.read_csv('2020_Competition_Training (1).csv', index_col=None, na_values=['N

print("Data Imported")
```

```
Data Imported
```

```
# Generalized Information of the Dataset.
df = data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65468 entries, 0 to 65467
Columns: 826 entries, person_id_syn to submcc_rsk_chol_ind
dtypes: float64(756), int64(48), object(22)
memory usage: 412.6+ MB
```

```
# Glimpse of top 5 entires of the dataset.
df.head()
```

	person_id_syn	transportation_issues	src_platform_cd	sex_cd	est_a
0	0002MOb79ST17bLYAe46eIc2	0	EM	F	
1	0004cMOS6bTLf34Y7AIca8f3	0	EM	F	
2	000536M9O3ST98LaYaeA29Ia	1	EM	F	
3	0009bMO9SfTLYe77A51I4ac3	0	EM	M	
4	000M7OeS66bTL8bY89Aa16Ie	0	EM	M	

5 rows × 826 columns

```
# Glimpse of last 5 entires of the dataset.
df.tail()
```

	person_id_syn	transportation_issues	src_platform_cd	sex_cd	e
65463	eMO26ST3c1L00bdfY42d90AI	0	EM	M	
65464	eMO279419S86bTcL4YAeI436	0	EM	M	
65465	eMO27S065aTLde71Ye89fAfI	0	EM	M	
65466	eMO2S6b0TL3Ydbef99AIcc25	0	EM	F	
65467	eMO2STLd74Y5A002fI6c7129	1	EM	M	

5 rows × 826 columns

```
# Columns in the Dataset.
df.columns
```

```
Index(['person_id_syn', 'transportation_issues', 'src_platform_cd', 'sex_cd',
       'est_age', 'smoker_current_ind', 'smoker_former_ind', 'lang_spoken_cd',
       'mabh_seg', 'cci_score',
       ...
       'submcc_rar_scl_ind', 'rx_gpi2_74_ind', 'rx_gpi2_89_ind',
       'rx_gpi2_96_ind', 'submcc_rsk_obe_ind', 'rx_gpi2_22_ind',
       'submcc_rsk_synx_ind', 'submcc_rsk_coag_ind', 'submcc_rsk_othr_ind',
       'submcc_rsk_chol_ind'],
      dtype='object', length=826)
```

```
# The dimension of the Dataset is 69572 observations with 826 variables.
df.shape
```

```
(65468, 826)
```

```
# Checking the NULL (missing values) columns from the dataset.
df.columns[df.isnull().any()]
```

```
df.columns[df.isnull().any()]
```

```
Index(['cms_ma_risk_score_nbr', 'cms_partd_ra_factor_amt',
       'cms_ra_factor_type_cd', 'cms_risk_adj_payment_rate_a_amt',
       'cms_risk_adj_payment_rate_b_amt', 'cms_risk_adjustment_factor_a_amt',
       'cms_rx_risk_score_nbr', 'cms_tot_ma_payment_amt',
       'cms_tot_partd_payment_amt', 'cons_cmys',
       ...
       'submcc_rar_scl_ind', 'rx_gpi2_74_ind', 'rx_gpi2_89_ind',
       'rx_gpi2_96_ind', 'submcc_rsk_obe_ind', 'rx_gpi2_22_ind',
       'submcc_rsk_synx_ind', 'submcc_rsk_coag_ind', 'submcc_rsk_othr_ind',
       'submcc_rsk_chol_ind'],
      dtype='object', length=755)
```

```
# Checking missing values are present or not
df.isnull().values.any()
```

```
True
```

```
# Summary of the dataset
df.describe(include='all')
```

	person_id_syn	transportation_issues	src_platform_cd	sex_cd	
count	65468	65468.000000	65468	65468	6
unique	65468	NaN	2	2	
top	M06006dOSTa0462c2L5YA9fI	NaN	EM	F	
freq	1	NaN	47030	38670	
mean	NaN	0.146850	NaN	NaN	
std	NaN	0.353959	NaN	NaN	
min	NaN	0.000000	NaN	NaN	
25%	NaN	0.000000	NaN	NaN	
50%	NaN	0.000000	NaN	NaN	
75%	NaN	0.000000	NaN	NaN	
max	NaN	1.000000	NaN	NaN	

11 rows × 826 columns

```
# Counting number of missing values
countofnulls = df.isnull().sum().sum()
countofnulls
```

```
483304
```

```
# Counting total elements in the dataset
totalelements = df.count().sum()
totalelements
```

```
53593264
```

```
# There are total 0.9% missing values in the dataset
percentageofnulls=(countofnulls/totalelements) * 100
percentageofnulls
```

```
0.9017998978379074
```

```
# Checking any duplicates observations in the dataset
duplicateRowsDF = df[df.duplicated()]
print("Duplicate Rows except first occurrence based on all columns are :")
print(duplicateRowsDF)
```

```
Duplicate Rows except first occurrence based on all columns are :
Empty DataFrame
Columns: [person_id_syn, transportation_issues, src_platform_cd, sex_cd, est_age
Index: []

[0 rows x 826 columns]
```

```
# Finding the unique values in transportation_issues variables
# 0 - Having no transportation issues
# 1 - Having transportation issues
pd.unique(df.transportation_issues)
```

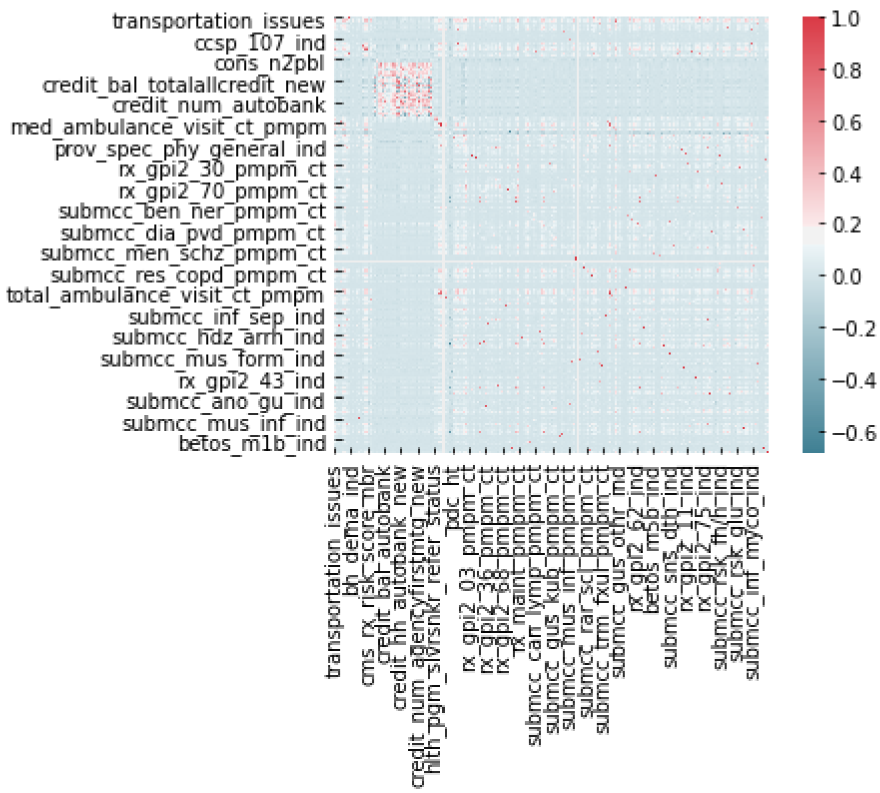
```
array([0, 1])
```

```
# Correlation values
df.corr(method='pearson')
```

	transportation_issues	est_age	smoker_current_ind	smoker_fo
transportation_issues	1.000000	-0.183735	0.098336	
est_age	-0.183735	1.000000	-0.170288	
smoker_current_ind	0.098336	-0.170288	1.000000	
smoker_former_ind	-0.015190	0.065703	-0.162310	
cci_score	-0.010585	0.408138	0.033061	

```
# Visulaizing the correlation
corr = df.corr(method='pearson')
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb9bf875898>
```



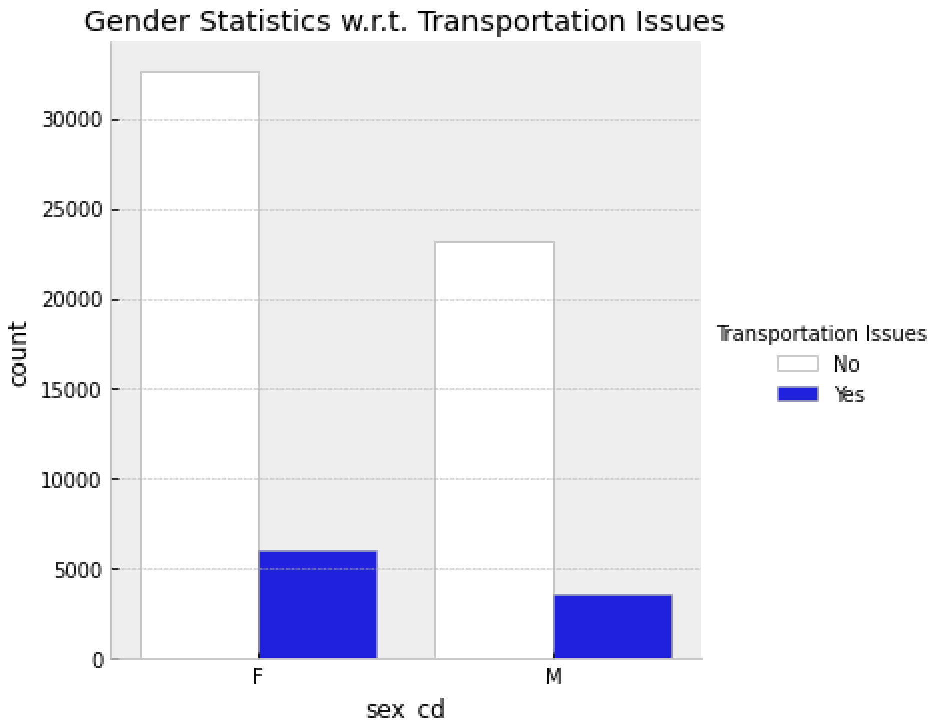

```
# Visulaizing the gender statistics with respect to target variable(transportation_is
g = sns.catplot(x="sex_cd", hue="transportation_issues", kind="count",
            palette="flag", edgecolor=".6",
            data=df)

# title
new_title = 'Transportation Issues'
g._legend.set_title(new_title)
# replace labels
new_labels = ['No', 'Yes']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)

import matplotlib.pyplot as plt
```

```
plt.title('Gender Statistics w.r.t. Transportation Issues')
```

```
Text(0.5, 1.0, 'Gender Statistics w.r.t. Transportation Issues')
```




```
# Information about zipcodes and their respective transportation issues.
pd.crosstab(df.zip_cd,df.transportation_issues)
```

transportation_issues 0 1

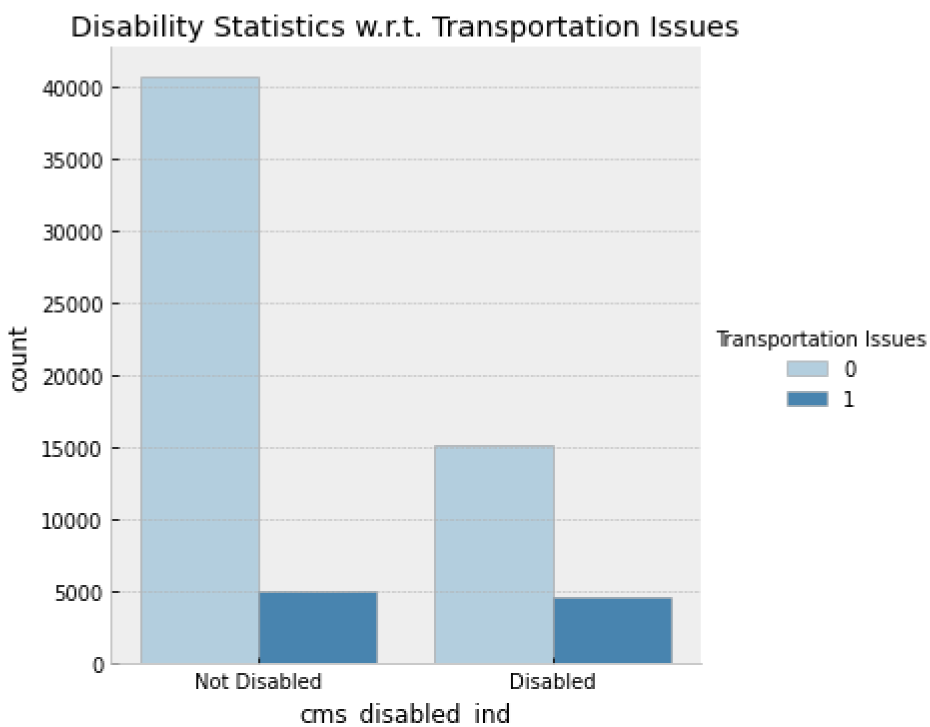
```
# Visulaizing the Disability statistics with respect to target variable(transportatio
p = sns.catplot(x="cms_disabled_ind", hue="transportation_issues", kind="count",
            palette="Blues", edgecolor=".6",
            data=df)

# title
new_title = 'Transportation Issues'
p._legend.set_title(new_title)
# replace labels
new_labels = ['No', 'Yes']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)
p.set_xticklabels( ('Not Disabled', 'Disabled') )

import matplotlib.pyplot as plt

plt.title('Disability Statistics w.r.t. Transportation Issues')
```

<seaborn.axisgrid.FacetGrid at 0x7fb9d28e1240>Text(0.5, 1.0, 'Disability Statist




```
# Extracting columns with Float datatype.

df_float = df.select_dtypes(include=[np.float])

df_float.head
```

```
<bound method NDFrame.head of        cci_score  betos_d1c_pmpm_ct  ...  submcc_r
0            3.0                0.0  ...                0.0         0
1            1.0                0.0  ...                0.0         1
2            3.0                0.0  ...                0.0         1
3            3.0                0.0  ...                0.0         1
4            3.0                0.0  ...                0.0         1
...          ...                ...  ...                ...         .
65463        4.0                0.0  ...                0.0         1
65464        7.0                0.0  ...                0.0         1
65465        4.0                0.0  ...                0.0         1
65466        6.0                0.0                     0.0         1
```

```
# Fill missing values with median column values of float
df_float_impute = df_float.fillna(df_float.median())

# Checking if the imputations of missing value is implemented correctly.
df_float_impute.isnull().values.any()
```

```
False
```

```
# Extracting columns with Integer datatype.

df_int64 = df.select_dtypes(include=[np.int64])

df_int64.head
```

```
<bound method NDFrame.head of        transportation_issues  ...  cmsd2_sns_gener
0                          0  ...                        1
1                          0  ...                        0
2                          1  ...                        1
3                          0  ...                        0
4                          0  ...                        0
...                      ...  ...                      ...
65463                      0  ...                        1
65464                      0  ...                        0
65465                      0  ...                        0
65466                      0  ...                        0
65467                      1  ...                        0

[65468 rows x 48 columns]>
```

```
# Fill missing values with median column values of interger datatype.

df_int64_impute = df_int64.fillna(df_int64.median())

# Checking if the imputations of missing value is implemented correctly.
df_int64_impute.isnull().values.any()
```

```
False
```

```
# Extracting columns with Object datatype.
```

```
df_object = df.select_dtypes(include=[np.object])

# Eliminating the Person_ID column from the Dataframe.
df_object = df_object.drop('person_id_syn', axis=1)

df_object.head
```

```
<bound method NDFrame.head of       src_platform_cd sex_cd lang_spoken_cd  ... z
0                  EM      F            ENG  ...  other   other    other
1                  EM      F            ENG  ...  other   other    other
2                  EM      F            ENG  ...  other   other    other
3                  EM      M            ENG  ...  other   other    other
4                  EM      M            ENG  ...  other   other    other
...               ...    ...            ...  ...    ...     ...      ...
65463              EM      M            ENG  ...  other   other    other
65464              EM      M            ENG  ...  other   other    other
65465              EM      M            ENG  ...  other   other    other
65466              EM      F            ENG  ...  33004     011       FL
65467              EM      M            ENG  ...    NaN     NaN      NaN

[65468 rows x 21 columns]>
```

```
# Fill missing values with most common element from column values of object datatype.

df_object_commonImpute = df_object.apply(lambda df_object: df_object.fillna(df_object

# Checking if the imputations of missing value is implemented correctly.
df_object_commonImpute.isnull().values.any()
```

```
False
```

```
# Factorizing the Catagorical Variables.

df_dummies = df_object_commonImpute.apply(lambda x: pd.factorize(x)[0])
df_dummies.head
```

```
<bound method NDFrame.head of        src_platform_cd  sex_cd  lang_spoken_cd  ..
0                    0       0               0  ...       0        0         0
```

```
#df_dummies = pd.get_dummies(df_object_commonImpute, drop_first=True)
#df_dummies.shape

# List of your dataframes to concat them together.
pdList = [df_dummies, df_int64_impute, df_float_impute]
new_df = pd.concat(pdList, axis=1)

new_df.isnull().values.any()
```

```
False
[65468 rows x 21 columns]>
```

```
# Checking the Dimension of Combined dataset.
new_df.shape
```

```
(65468, 825)
```

```
# Splitting the dataset into train and validation sets
# Training set- 80% of original dataset
# Validation set- 20% of original dataset.

from sklearn.model_selection import train_test_split
X = new_df.drop ('transportation_issues', axis=1)
Y = new_df['transportation_issues']
X_train, X_Valid, Y_train, Y_Valid = train_test_split( X, Y, test_size=0.2, random_sta

print('X Training Dataset Shape: ', X_train.shape)
print('Y Training Dataset Shape: : ', Y_train.shape)

print('X Validation Dataset Shape: ', X_Valid.shape)
print('Y Validation Dataset Shape: : ', Y_Valid.shape)
```

```
X Training Dataset Shape:  (52374, 824)
Y Training Dataset Shape: :  (52374,)
X Validation Dataset Shape:  (13094, 824)
Y Validation Dataset Shape: :  (13094,)
```

```
# The first TRIAL Model training using Logistic Regression.

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

model = LogisticRegression(solver='liblinear', random_state=0)
model.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
```

```
# Looking at the Confusion Matrix for the training Dataset.
# Here we could see the model gives accuracy of 85.4%.
confusion_matrix(Y_train, model.predict(X_train))
```

```
array([[44329,   354],
       [ 7300,   391]])
```

```
# Evaluating the model with Validation Dataset.
Y_predict = model.predict(X_Valid)
```

```
# Looking at the Confusion Matrix for the Validation Dataset.
# Here we could see the model gives accuracy of 85.3% on validation Dataset.

confusion_matrix(Y_Valid, model.predict(X_Valid))
```

```
array([[11090,    81],
       [ 1842,    81]])
```

```
# Classification result with respect to Validation Dataset and checking the modelling

print(classification_report(Y_Valid, Y_predict))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92     11171
           1       0.50      0.04      0.08      1923

    accuracy                           0.85     13094
   macro avg       0.68      0.52      0.50     13094
weighted avg       0.81      0.85      0.80     13094
```

Things to be done:

1. Converting Catagorical attributes into dummy variables.
2. Apply Feature selection techniques.
3. Balancing the dataset.
4. Normalization of the dataset.