Car Rollover Prediction Report (Assignment 2)

Lecture: CS452/552 – Data Science with Python

University: Ozyegin University

Assignment Prepared by: Furkan Cantürk

Instructor of the Lecture: Assistant Prof. Reyhan Aydoğan

Tuna Tuncer – S018474

1. Summary/Abstract

This report talks about the different models created for the smart pricing system that should be implemented for a car insurance company, the datasets used and the process to create the models. In order to create an intelligent pricing system, it was decided to conduct data analysis on accidents. NHTSA's FARS (Fatality Analysis Reporting System) was chosen as the dataset to be used. Among the FARS datasets, two different datasets, *person* and *vindecode*, were selected. The *rollovername* in the person dataset was chosen as the feature to be predicted. This feature keeps the information whether the vehicle has rolled over as a result of the accident. If the vehicle and driver information can be used to predict whether the vehicle will roll over in the accident or not, this will also lead to a conclusion about how much damage the vehicle is in danger of. As a result, the incoming customer can be priced wisely by using the risk of damage to the vehicle.

2. Data Loading and Cleaning

In this part, two different datasets (Person & Vindecode) about the accident, the vehicles and people involved in the accident will be fetched. Afterwards, the data will be cleaned by dropping unnecessary features. Finally, a single dataset will be created by combining these two datasets.

2.1. Fetching the Datasets

Person and Vindecode(Vehicle) datasets of 2014-2019 are fetched from the API provided. While the shape of the person data set is **(498002, 131)**, the shape of the vehicle data set is **(313877, 105)**.

2.2. Determining and Dropping Useless Columns

First, before starting to clean the features, rows of the *per_typname* feature that did not have a value of "Driver of a Motor Vehicle In-Transport" were dropped because only drivers will be needed in this column when merging the person and vehicle datasets in the part 2.3. This operation caused the shape of the person data frame to drop to **(304126, 131)**.

Cleaning the features was first started by getting rid of the encoded features. For example, there were two different features called *state* and *statename*, which were encoded and decoded versions of each other. In order not to lose its semantic meaning, the encoded (*state*) feature has been dropped. After the encoded columns were dropped, the shape of the person data frame was reduced to **(304126, 71)**, while the shape of the vehicle data frame was reduced to **(313877, 70)**.

Subsequently, columns with missing values for more than ninety percent of both data frames were dropped and this operation reduced the shape of the person and the vehicle data frame to **(304126, 71)** and **(313877, 59)** respectively.

Afterwards, only those whose columns have single values from both data frames are dropped. As a result, the shapes of person and vehicle data frames became **(304126, 65)** and **(313877, 57)**, respectively.

Finally, the features that can be obtained during or after the crash were selected and dropped from both data frames. As a result, the shapes of the person and vehicle data frames became **(304126, 9)** and **(313877, 57)**, respectively.

In conclusion, **122** columns were dropped from the person data frame, while **48** columns were dropped from the vehicle data frame.

2.3. Merging Vehicle and Person Dataset

In this part, person and vehicle data frame were joined using *caseyear*, *statename*, *st_case*, *veh_no* features and a new data frame with shape **(304126, 62)** was created. All operations to be performed in the rest of the report will be applied on this data frame.

Column names of the merged data frame: [*'abs_t'*, *'agename'*, *'battyp_t'*, *'blocktype'*, *'bodystyl_t'*, *'carbtype_t'*, *'caseyear'*, *'countyname'*, *'cylndrs'*, *'dispclmt'*, *'displci'*, *'doors'*, *'drivetyp_t'*, *'drivwhls'*, *'drl_t'*, *'enghead_t'*, *'engmfg_t'*, *'engmodel'*, *'engvincd'*, *'engvvt'*, *'fuel_t'*, *'fuelinj_t'*, *'gvwrange_t'*, *'incomplt'*, *'mfg_t'*, *'msrp'*, *'origin_t'*, *'per_no'*, *'plntcity'*, *'plntctry_t'*, *'plntstat_t'*, *'rearsize_t'*, *'rollovername'*, *'rstrnt_t'*, *'security_t'*, *'segmnt_t'*, *'sexname'*, *'shipweight'*, *'st_case'*, *'statename'*, *'supchrgr_t'*, *'tiresz_f_t'*, *'tkaxlef_t'*, *'tkaxler_t'*, *'tkbedl_t'*, *'tkbrak_t'*, *'tkcab_t'*, *'tkduty_t'*, *'tonrating'*, *'turbo_t'*, *'veh_no'*, *'vehtype_t'*, *'vinmake_t'*, *'vinmodel_t'*, *'vintrim1_t'*, *'vintrim_t'*, *'vinyear'*, *'vlvclndr'*, *'vlvtotal'*, *'wheels'*, *'whlblg'*, *'whlbsh'*]

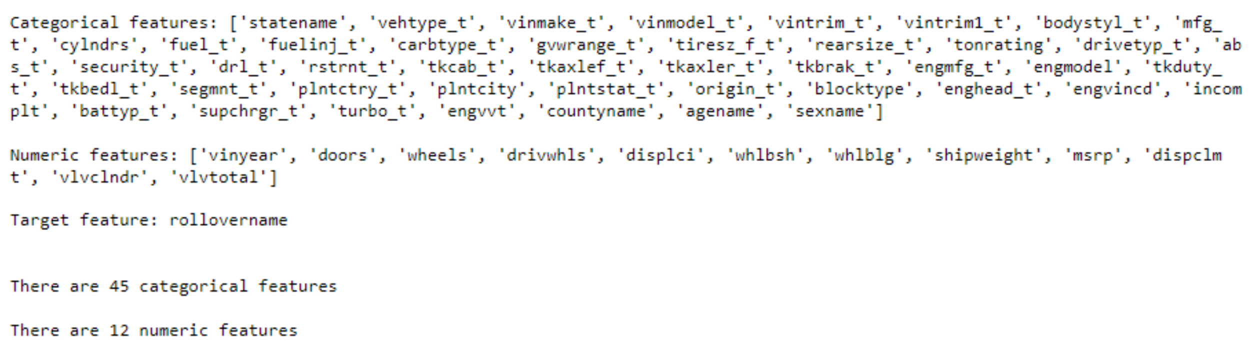
3. Exploratory Data Analysis (EDA)

This section contains the characteristics of categorical and numerical features in general, as well as statistics that may be beneficial in data cleaning and feature engineering. Additionally, the target feature's possible values are reported, and the problem's bounds have been established.

3.1. Detecting Missing Values

First, all features except *caseyear*, *st_case*, and *veh_no* were divided into three groups as *categorical*, *numeric*, and *target_variable*. These three attributes are not included in any of the groups, since they are only identifier columns used to join the person and vehicle datasets.

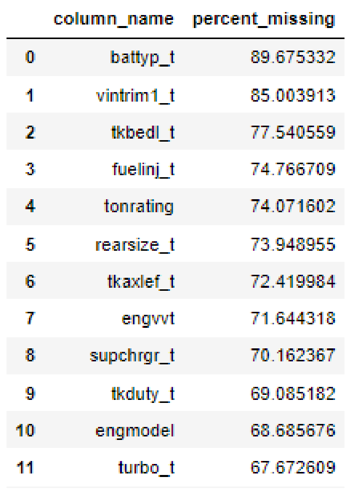
As it can be observed in Figure 1, the dataframe contains 45 categorical and 13 numeric features.



```
Categorical features: ['statename', 'vehtype_t', 'vinmake_t', 'vinmodel_t', 'vintrim_t', 'vintrim1_t', 'bodystyl_t', 'mfg_
t', 'cylndrs', 'fuel_t', 'fuelinj_t', 'carbtype_t', 'gvwrange_t', 'tiresz_f_t', 'rearsize_t', 'tonrating', 'drivetyp_t', 'ab
s_t', 'security_t', 'drl_t', 'rstrnt_t', 'tkcab_t', 'tkaxlef_t', 'tkaxler_t', 'tkbrak_t', 'engmfg_t', 'engmodel', 'tkduty_
t', 'tkbedl_t', 'segmnt_t', 'plntctry_t', 'plntcity', 'plntstat_t', 'origin_t', 'blocktype', 'enghead_t', 'engvincd', 'incom
plt', 'battyp_t', 'supchrgr_t', 'turbo_t', 'engvvt', 'countyname', 'agename', 'sexname']

Numeric features: ['vinyear', 'doors', 'wheels', 'drivwhls', 'displci', 'whlbsh', 'whlblg', 'shipweight', 'msrp', 'dispclm
t', 'vlvclndr', 'vlvtotal']

Target feature: rollovername


There are 45 categorical features

There are 12 numeric features
```

[1] Categorical, numeric and target features

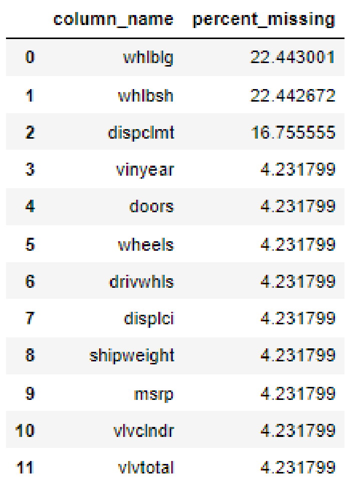
Apart from that, it was calculated what percentage of each feature contains null values. In Figure 2, it is seen that many of the categorical features contain a high rate of missing values.



	column_name	percent_missing
0	battyp_t	89.675332
1	vintrim1_t	85.003913
2	tkbedl_t	77.540559
3	fuelinj_t	74.766709
4	tonrating	74.071602
5	rearsize_t	73.948955
6	tkaxlef_t	72.419984
7	engvvt	71.644318
8	supchrgr_t	70.162367
9	tkduty_t	69.085182
10	engmodel	68.685676
11	turbo_t	67.672609

[2] Percentage of missing values of the first 11 categorical columns

Numeric features, on the other hand, have a substantially smaller rate of missing values than categorical features, as seen in Figure 3.



	column_name	percent_missing
0	whlblg	22.443001
1	whlbsh	22.442672
2	dispclmt	16.755555
3	vinyear	4.231799
4	doors	4.231799
5	wheels	4.231799
6	drivwhls	4.231799
7	displci	4.231799
8	shipweight	4.231799
9	msrp	4.231799
10	vlvclndr	4.231799
11	vlvtotal	4.231799

[3] Percentage of missing values of the 11 numeric columns

3.2. Dropping Duplicate and Empty Rows/Columns

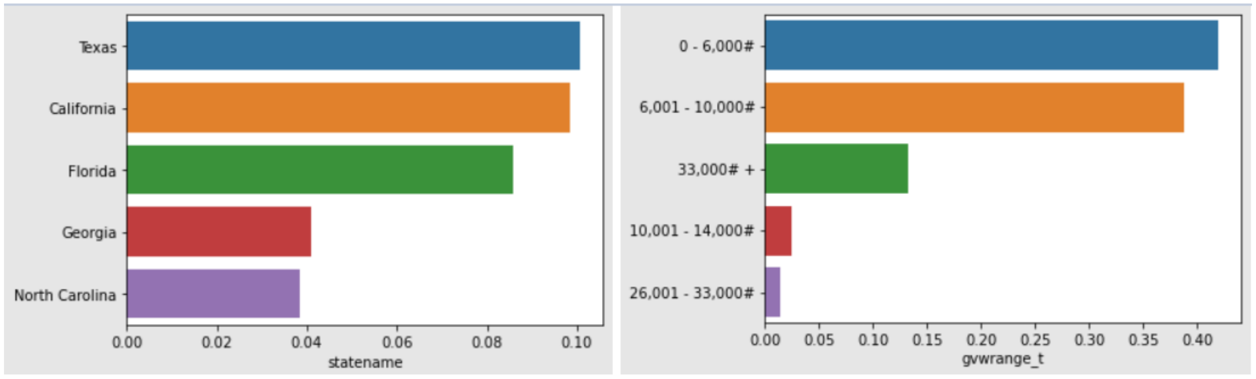
In the data frame, duplicate rows were examined, but no entries were discovered to be duplicated.

The features *caseyear*, *st_case*, *veh_no*, and *per_no* were removed since they are utilized as identifiers and do not contribute to the target variable during the estimation process. After this process, the shape of the data frame has decreased to **(304126, 58)**.

In addition, because the cleaning of the columns is not completely finished, and in order not to lose important information by shrinking the data at hand, no action was taken on the rows.

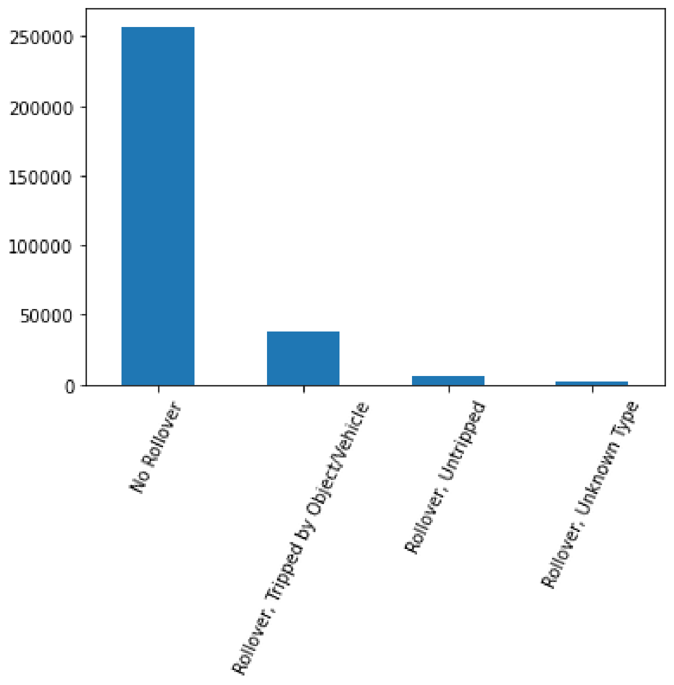
3.3. Graphical Interpretations and Value Counts

In this section, the features are made more concrete using the help of graphics, making it easier to grasp insights about them. The two sample graphs seen in Figure 4 were applied to all categorical features to show the most frequent 5 values of the associated feature. Since some categorical features have too many unique values, the most frequent 5 values of all features were represented in the graphs.



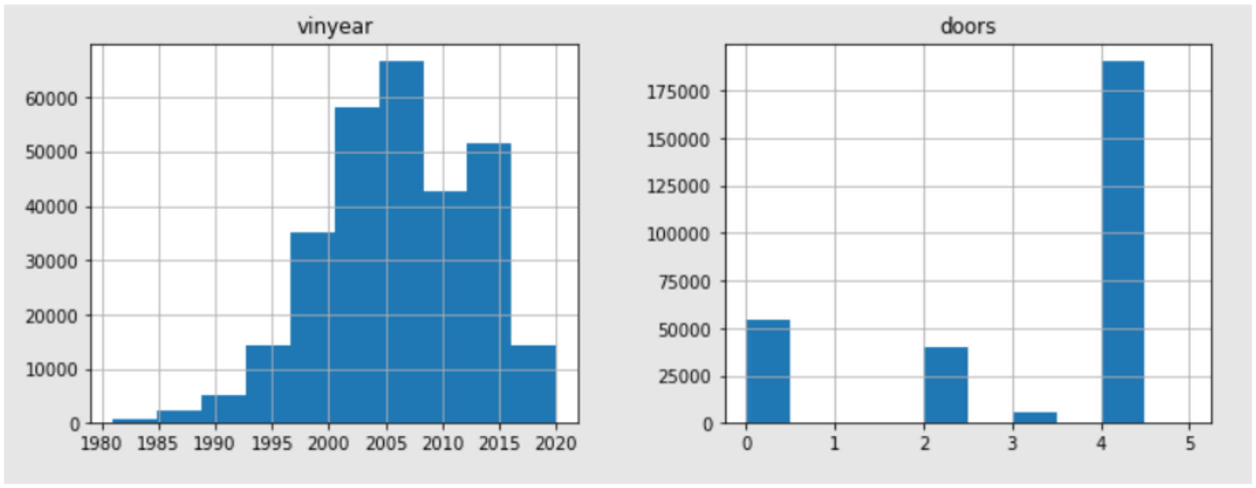
[4] Bar plots of the value frequencies of *statename* and *gvwrange_t* (categorical) features

When looking at the graph in Figure 5, it's visible that the target feature has four distinct values, which suggests multi-label classification. However, when looking at the four values, it's obvious that one of them can be categorized as no rollover and the others as rollover. This indicates that the problem is actually **binary classification**. Another crucial factor to remember is that the no rollover value is significantly higher than the rollover value. Because of this imbalance, no rollover may be given higher priority during the model's predictions.



[5] Bar plot of unique values of the target feature

The two example histograms seen in Figure 6 are plotted for all numeric features. Thus, the distribution of each numeric feature is roughly visualized.

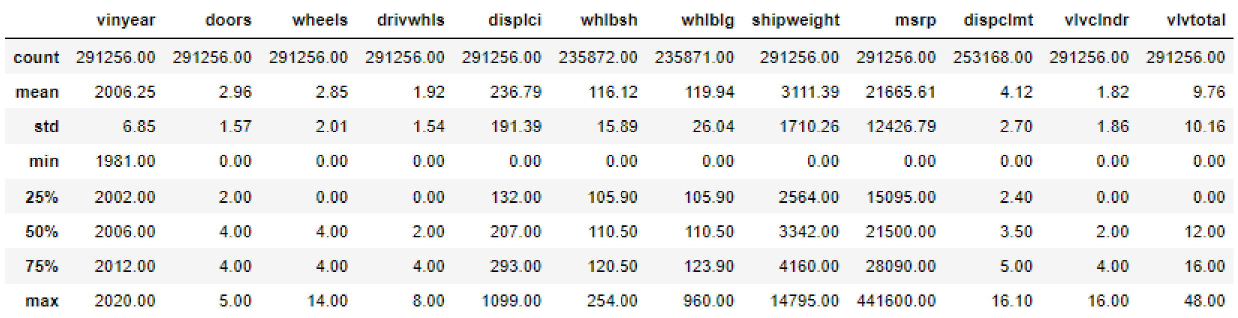


[6] Histograms of *vinyear* and *doors* (numeric) features

3.4. Statistical Tables

Figure 7 shows some statistical values such as count, mean, min, etc. for all numeric features. The statistics in this table will be very useful during the imputation of missing numeric features. However, when examined carefully, as an example, it is seen that the min value of the wheels feature is 0, but no vehicle's wheels can be 0, so it can be thought that 0 values in some features mean null. Unfortunately, null values used as 0 also cause all statistics in the table to be incorrect. Therefore, zeros that mean null should be avoided.

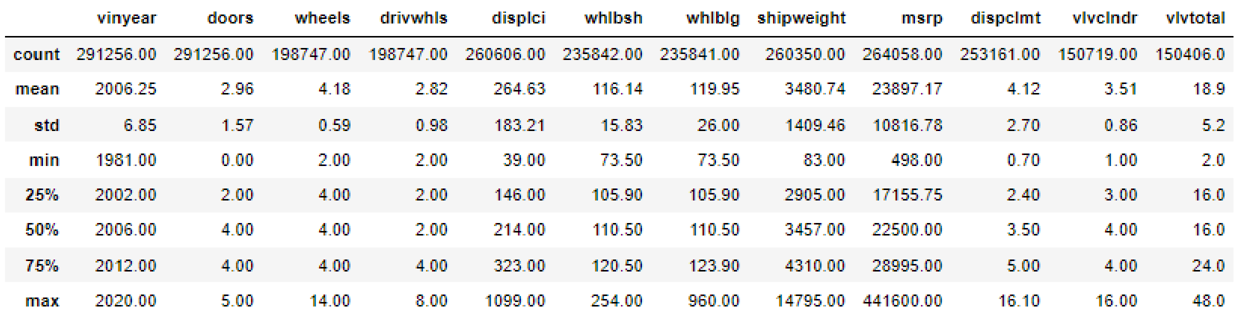
Therefore, in this section, 0 values in numeric features other than *vinyear* and *doors* are replaced with *np.NaN*.



	vinyear	doors	wheels	drivwhls	displci	whlbsh	whlblg	shipweight	msrp	dispclmt	vlvclndr	vlvtotal
count	291256.00	291256.00	291256.00	291256.00	291256.00	235872.00	235871.00	291256.00	291256.00	253168.00	291256.00	291256.00
mean	2006.25	2.96	2.85	1.92	236.79	116.12	119.94	3111.39	21665.61	4.12	1.82	9.76
std	6.85	1.57	2.01	1.54	191.39	15.89	26.04	1710.26	12426.79	2.70	1.86	10.16
min	1981.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	2002.00	2.00	0.00	0.00	132.00	105.90	105.90	2564.00	15095.00	2.40	0.00	0.00
50%	2006.00	4.00	4.00	2.00	207.00	110.50	110.50	3342.00	21500.00	3.50	2.00	12.00
75%	2012.00	4.00	4.00	4.00	293.00	120.50	123.90	4160.00	28090.00	5.00	4.00	16.00
max	2020.00	5.00	14.00	8.00	1099.00	254.00	960.00	14795.00	441600.00	16.10	16.00	48.00

[7] (Incorrect) Statistical information table for numeric values (due to null values)

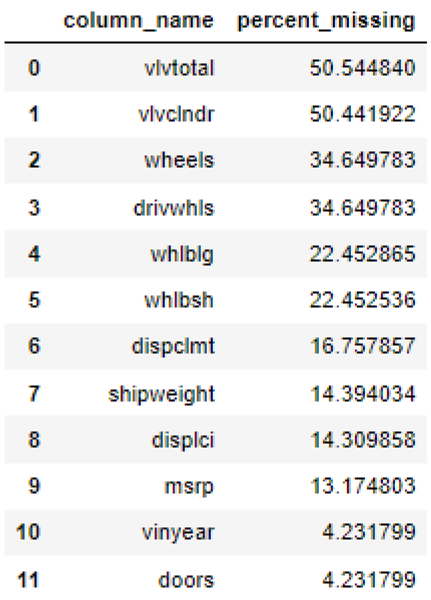
After this change, the new state of the statistics table is as in figure 8. As can be observed, the mean values have increased while the min value has changed.



	vinyear	doors	wheels	drivwhls	displci	whlbsh	whlblg	shipweight	msrp	dispclmt	vlvclndr	vlvtotal
count	291256.00	291256.00	198747.00	198747.00	260606.00	235842.00	235841.00	260350.00	264058.00	253161.00	150719.00	150406.0
mean	2006.25	2.96	4.18	2.82	264.63	116.14	119.95	3480.74	23897.17	4.12	3.51	18.9
std	6.85	1.57	0.59	0.98	183.21	15.83	26.00	1409.46	10816.78	2.70	0.86	5.2
min	1981.00	0.00	2.00	2.00	39.00	73.50	73.50	83.00	498.00	0.70	1.00	2.0
25%	2002.00	2.00	4.00	2.00	146.00	105.90	105.90	2905.00	17155.75	2.40	3.00	16.0
50%	2006.00	4.00	4.00	2.00	214.00	110.50	110.50	3457.00	22500.00	3.50	4.00	16.0
75%	2012.00	4.00	4.00	4.00	323.00	120.50	123.90	4310.00	28995.00	5.00	4.00	24.0
max	2020.00	5.00	14.00	8.00	1099.00	254.00	960.00	14795.00	441600.00	16.10	16.00	48.0

[8] (Correct) Statistical information table for numeric values

Figure 9 shows a repainted version of the table in figure 3 since the adjustment made with null values impacts it. When the two tables are compared, it is clear that the null percentages have increased.



	column_name	percent_missing
0	vlvtotal	50.544840
1	vlvclndr	50.441922
2	wheels	34.649783
3	drivwhls	34.649783
4	whlblg	22.452865
5	whlbsh	22.452536
6	dispclmt	16.757857
7	shipweight	14.394034
8	displci	14.309858
9	msrp	13.174803
10	vinyear	4.231799
11	doors	4.231799

[9] Percentage of missing values of the 11 numeric columns

4. Data Cleaning and Feature Engineering

4.1. Handling Missing Values (Dropping and/or Imputation)

Missing values in numeric and categorical features are imputed in this section. No features were dropped in this part, as features that were already more than 90% null had previously been dropped on part 2.2.

Numeric features *vinyear*, *doors*, *wheels*, *drivwhls*, *vlvclndr* and *vlvtotal* were imputed for the most frequent values in their corresponding columns, while *displci*, *whlbsh*, *whlblg*, *shipweight*, *msrp*, *dispclmt*, and *displci* features were imputed for the mean values.

The reason for choosing the most frequent value technique for imputation is that the features used can only take integer (discrete) values. Therefore, the mean imputation technique cannot be used for them because the mean values will very likely not be integers. On the other hand, the reason for choosing the mean technique for imputation is that it fits well with the data.

For categorical features, the most frequent value technique is used for imputation because it is not possible to get the mean of a categorical data.

As a consequence, by the end of this section, all missing values have been imputed, and the data frame had no missing values remaining.

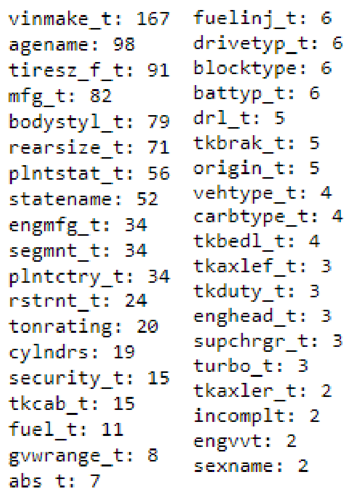
4.2. Reducing Cardinality of Categorical Features

In order to train a model, all features must be numeric features. Numeric features were prepared by cleaning in previous parts, and in this part, categorical features will be made ready to be converted into numeric features with cardinality reduction. However, before starting cardinality reduction, it was decided to drop 7 different categorical features.

- It has been decided to drop the *countyname* (value count=2979) feature because it will be very difficult to separate into different groups and the *statename* feature provides the necessary information already.
- It has been decided to drop the *engvincd* (value count=2504) feature because it will be very difficult to separate into different groups and it is not believed to be a feature that affects the target feature much.
- It has been decided to drop the *vintrim_t* (value count=2124) feature because it will be very difficult to separate into different groups and it is not believed to be a feature that affects the target feature much.
- It has been decided to drop the *vinmodel_t* (value count=2110) feature because it will be very difficult to separate into different groups and it is not believed to be a feature that affects the target feature much.
- It has been decided to drop the *plntcity* (value count=533) feature because it will be very difficult to separate into different groups and the *statename* feature provides the necessary information.
- It has been decided to drop the *vintrim1_t* (value count=360) feature because it will be very difficult to separate into different groups and it is not believed that trim of a vehicle will affect the target feature much.
- It has been decided to drop the *engmodel* (value count=237) feature because it will be very difficult to separate into different groups and it is not believed to be a feature that affects the target feature much.

By dropping these 7 features, the shape of the data frame will decrease to **(304126, 51)**.

In Figure 10, the value counts of categorical features can be seen before the cardinality reduction is applied. While these values decrease for some features after cardinality reduction, they will remain constant for others.



```
vinmake_t: 167   fuelinj_t: 6
agename: 98      drivetyp_t: 6
tiresz_f_t: 91   blocktype: 6
mfg_t: 82        battyp_t: 6
bodystyl_t: 79   drl_t: 5
rearsize_t: 71   tkbrak_t: 5
plntstat_t: 56   origin_t: 5
statename: 52    vehtype_t: 4
engmfg_t: 34     carbtype_t: 4
segmnt_t: 34     tkbedl_t: 4
plntctry_t: 34   tkaxlef_t: 3
rstrnt_t: 24     tkduty_t: 3
tonrating: 20    enghead_t: 3
cylndrs: 19      supchrgr_t: 3
security_t: 15   turbo_t: 3
tkcab_t: 15      tkaxler_t: 2
fuel_t: 11       incomplt: 2
gvwrange_t: 8    engvvt: 2
abs_t: 7         sexname: 2
```

[10] Value counts of categorical features before cardinality reduction

Of the remaining 38 categorical features, it was decided that there was no need to apply reduction of cardinality to 17 of them because they were already simplified as much as possible.

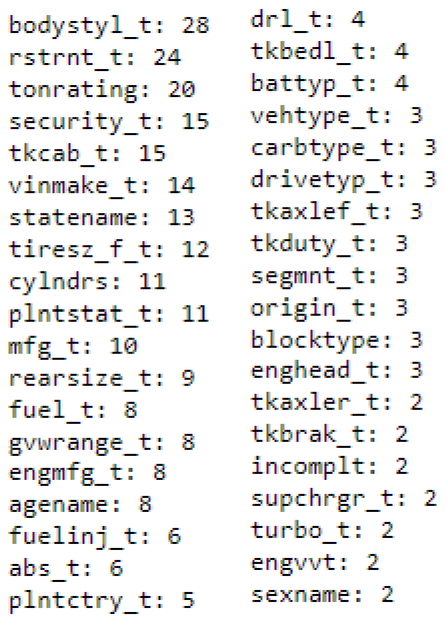
For the remaining 21 categorical features, their cardinality has been reduced by using 3 different techniques.

Firstly, values that have the same or similar meanings but written differently for 12 categorical features (*segmnt_t*, *cylndrs*, *bodystyl_t*, *vehtype_t*, *fuel_t*, *carbtype_t*, *turbo_t*, *drivetyp_t*, *abs_t*, *drl_t*, *supchrgr_t*, *origin_t*) were replaced and reduced to a single value. For example, the *battyp_t* feature contained 3 different values, 'Lithium-ion', 'Lithium-ion Polymer' and 'Lithium Ion' with the same or similar meanings. All of these values have been reduced to 'Lithium Ion'.

Another technique used for the 3 categorical features (*tiresz_f_t*, *rearsize_t*, *agename*) was to create a relation between the values and reduce the values to a smaller number of groups according to this relation. For example, the *agename* feature contained 98 unique values (unique age). Categories such as 0-20, 20-40, 40-60 years old were created and the corresponding values were reduced to these groups.

Third, for the remaining 6 categorical features (*vinmake_t, mfg_t, plntstat_t, statename, engmfg_t, plntctry_t*) that are difficult to reduce, a few very frequent values were taken and all the remaining values were named "other".

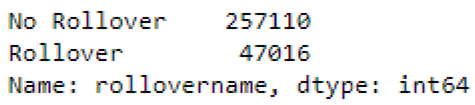
In Figure 11, it is seen that the value counts of categorical features have decreased considerably compared to the previous situation.



```
bodystyl_t: 28     drl_t: 4
rstrnt_t: 24       tkbedl_t: 4
tonrating: 20      battyp_t: 4
security_t: 15     vehtype_t: 3
tkcab_t: 15        carbtype_t: 3
vinmake_t: 14      drivetyp_t: 3
statename: 13      tkaxlef_t: 3
tiresz_f_t: 12     tkduty_t: 3
cylndrs: 11        segmnt_t: 3
plntstat_t: 11     origin_t: 3
mfg_t: 10          blocktype: 3
rearsize_t: 9      enghead_t: 3
fuel_t: 8          tkaxler_t: 2
gvwrange_t: 8      tkbrak_t: 2
engmfg_t: 8        incomplt: 2
agename: 8         supchrgr_t: 2
fuelinj_t: 6       turbo_t: 2
abs_t: 6           engvvt: 2
plntctry_t: 5      sexname: 2
```

[11] Value counts of categorical features after cardinality reduction

As mentioned in part 3.3, it is observed that the rollovername (target) feature is reduced to two values, no rollover and rollover, in Figure 12.



```
No Rollover    257110
Rollover        47016
Name: rollovername, dtype: int64
```

[12] Value counts of *rollovername* feature

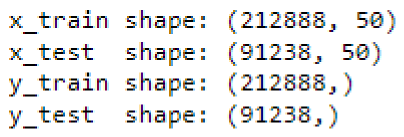
5. Data Splitting and Transformation

Instead of choosing the test set as 2019, I think it would be a more logical choice to distribute the data as 70 percent train and 30 percent test. Because if 2019 is chosen as the test set, there will be no data about 2019 in the training of the model. If there is a change in the crash dynamics of 2019 compared to previous years, this will negatively affect both the training and testing of the model. On the other hand, when the data is separated as 70 percent train and 30 percent test, the above-mentioned issue will not cause problems because there will be accident examples from 2019 in both the train and the test set.

5.1. Splitting the Data

The remaining 51 features were divided into 50 x and 1 y (*rollovername*). Subsequently, the x and y data frames were divided into 70 percent train and 30 percent test data frames.

Shapes of all train and test data frames created are shown in Figure 13.

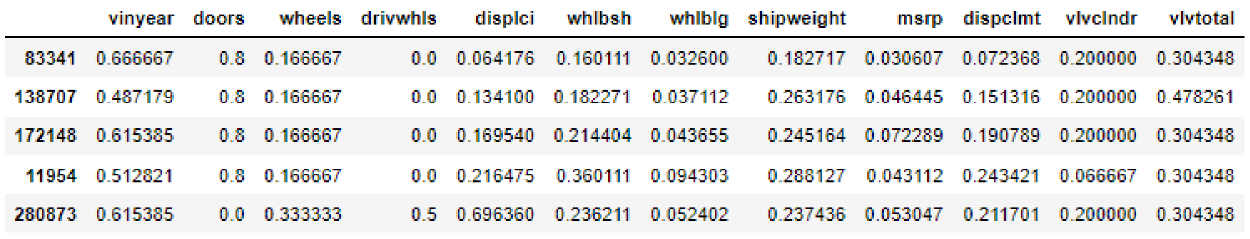


```
x_train shape: (212888, 50)
x_test  shape: (91238, 50)
y_train shape: (212888,)
y_test  shape: (91238,)
```

[13] Shapes of train and test data frames

5.2. Scaling the Numerical Features

In this part, firstly, numeric values were scaled using *StandardScaler*, but since chi2 was wanted to be used as score function while making feature selection in part 6, it was decided that it is wiser to use *MinMaxScaler* for scaling. Because chi2 scoring function does not accept negative values, numeric values are scaled to (0,1) range with *MinMaxScaler* as it also can be observed in the Figure 14.

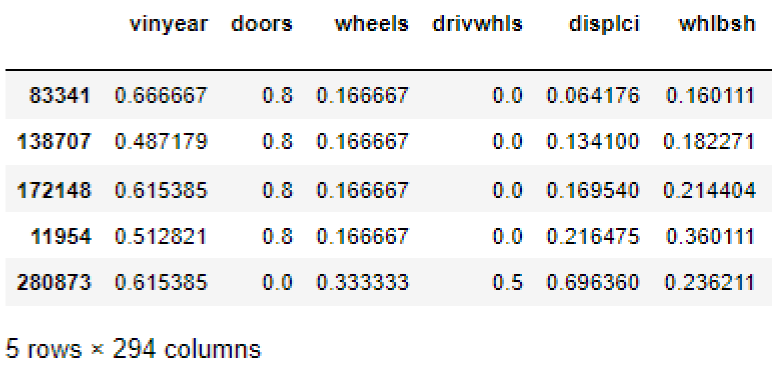


	vinyear	doors	wheels	drivwhls	displci	whlbsh	whlblg	shipweight	msrp	dispclmt	vlvclndr	vlvtotal
83341	0.666667	0.8	0.166667	0.0	0.064176	0.160111	0.032600	0.182717	0.030607	0.072368	0.200000	0.304348
138707	0.487179	0.8	0.166667	0.0	0.134100	0.182271	0.037112	0.263176	0.046445	0.151316	0.200000	0.478261
172148	0.615385	0.8	0.166667	0.0	0.169540	0.214404	0.043655	0.245164	0.072289	0.190789	0.200000	0.304348
11954	0.512821	0.8	0.166667	0.0	0.216475	0.360111	0.094303	0.288127	0.043112	0.243421	0.066667	0.304348
280873	0.615385	0.0	0.333333	0.5	0.696360	0.236211	0.052402	0.237436	0.053047	0.211701	0.200000	0.304348

[14] First five rows of the scaled numeric values of the train data frame

5.3. Encoding the Categorical Features

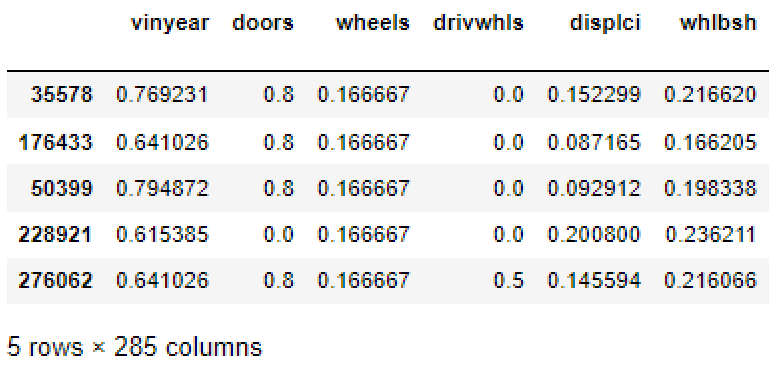
In this part, the data was first one hot encoded using the *get_dummies()* method in the *pandas* library, but an unexpected problem was encountered. As can be seen in Figures 15 and 16, the number of features of the train and test dataset is different. It is estimated that the reason for this is that not all feature values are evenly distributed in the test and train dataset before encoding. Since this will cause problems when testing the model in the future, one hot encoding has been decided to be done with the *OneHotEncoder* module of the *sklearn* library.



	vinyear	doors	wheels	drivwhls	displci	whlbsh
83341	0.666667	0.8	0.166667	0.0	0.064176	0.160111
138707	0.487179	0.8	0.166667	0.0	0.134100	0.182271
172148	0.615385	0.8	0.166667	0.0	0.169540	0.214404
11954	0.512821	0.8	0.166667	0.0	0.216475	0.360111
280873	0.615385	0.0	0.333333	0.5	0.696360	0.236211

5 rows × 294 columns

[15] First five rows and the shape of the encoded train data frame (using *get_dummies()*)

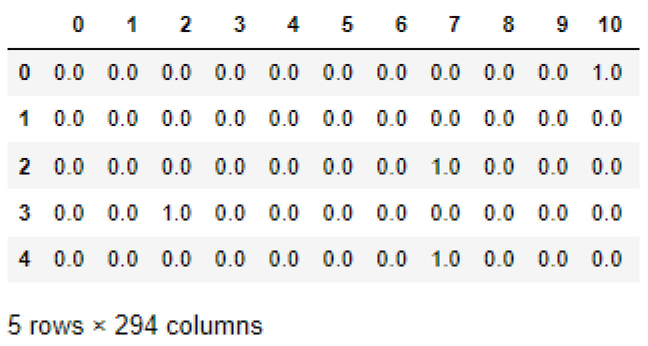


	vinyear	doors	wheels	drivwhls	displci	whlbsh
35578	0.769231	0.8	0.166667	0.0	0.152299	0.216620
176433	0.641026	0.8	0.166667	0.0	0.087165	0.166205
50399	0.794872	0.8	0.166667	0.0	0.092912	0.198338
228921	0.615385	0.0	0.166667	0.0	0.200800	0.236211
276062	0.641026	0.8	0.166667	0.5	0.145594	0.216066

5 rows × 285 columns

[16] First five rows and the shape of the encoded test data frame (using *get_dummies()*)

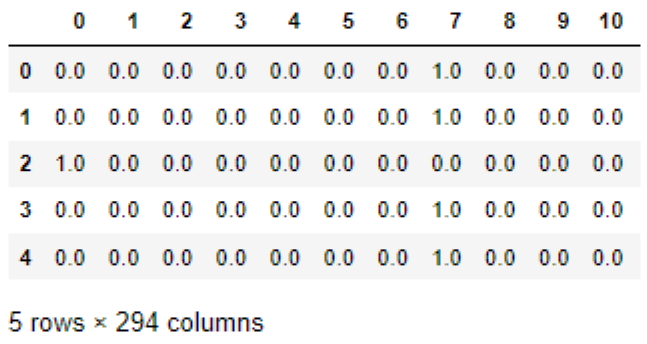
When categorical features were encoded using the *OneHotEncoder* module, the number of features of the train and test dataset was the same as seen in figures 17 and 18.



	0	1	2	3	4	5	6	7	8	9	10
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

5 rows × 294 columns

[17] First five rows and the shape of the encoded train data frame (using *OneHotEncoder*)

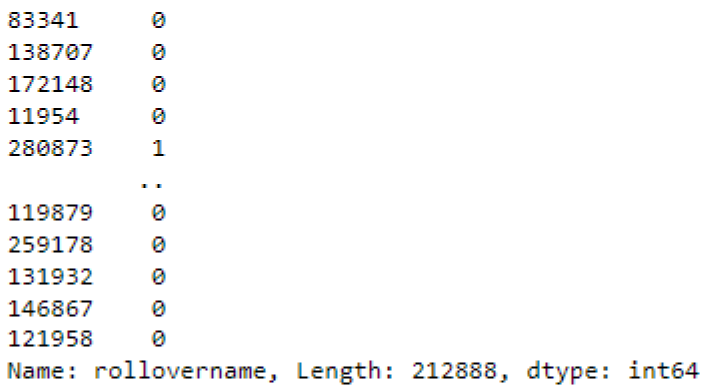


	0	1	2	3	4	5	6	7	8	9	10
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

5 rows × 294 columns

[18] First five rows and the shape of the encoded test data frame (using *OneHotEncoder*)

Also, target train and target test set were replaced with "No Rollover" to be 0 and "Rollover" to be 1 to prepare for model training.



```
83341     0
138707    0
172148    0
11954     0
280873    1
         ..
119879    0
259178    0
131932    0
146867    0
121958    0
Name: rollovername, Length: 212888, dtype: int64
```

[19] Replaced values of target train set.

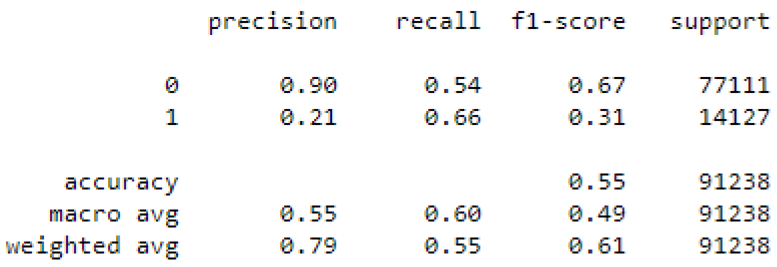
6. Feature Selection

In this part, 10 different features have been selected using the **select k best features** technique. Since the problem type is classification, **chi2** is used as score function. [1] Gajawada points out that chi2 test is used in statistics when measuring the independence between two events. Therefore, in this section, it has been decided to select the best 10 features by using the chi2 test to measure the independence of each feature with the target feature.

Indexes of the selected features are as follows: [13, 15, 22, 34, 63, 72, 77, 144, 145, 168]

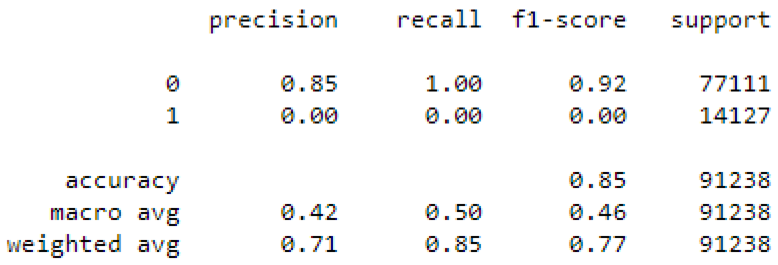
7. Training and Performance Evaluation

In this part, naive bayes, logistic regression, random forest and support vector machines models are trained. **Complement naive bayes** was chosen as the naive bayes model because it is more common to apply to imbalanced datasets. The classification reports generated for each trained model can be viewed below.



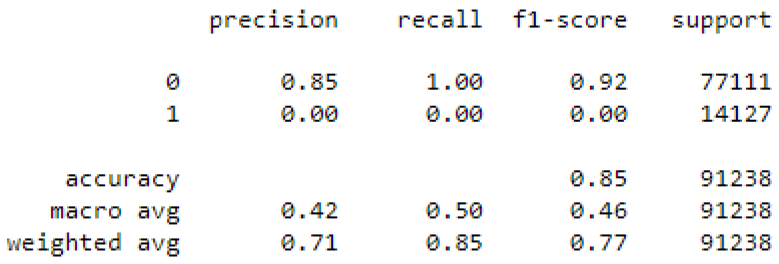
```
              precision    recall  f1-score   support

           0       0.90      0.54      0.67     77111
           1       0.21      0.66      0.31     14127

    accuracy                           0.55     91238
   macro avg       0.55      0.60      0.49     91238
weighted avg       0.79      0.55      0.61     91238
```

[20] Classification report for Complement Naïve Bayes model



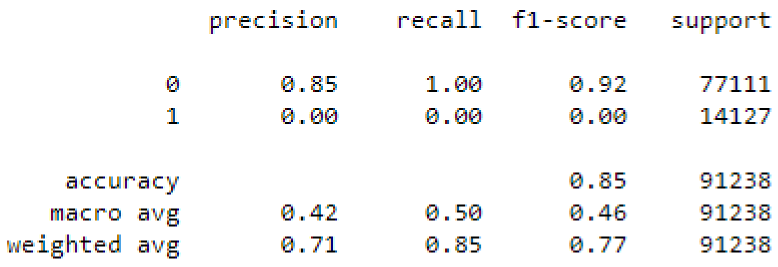
```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92     77111
           1       0.00      0.00      0.00     14127

    accuracy                           0.85     91238
   macro avg       0.42      0.50      0.46     91238
weighted avg       0.71      0.85      0.77     91238
```

[21] Classification report for Logistic Regression model



```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92     77111
           1       0.00      0.00      0.00     14127

    accuracy                           0.85     91238
   macro avg       0.42      0.50      0.46     91238
weighted avg       0.71      0.85      0.77     91238
```

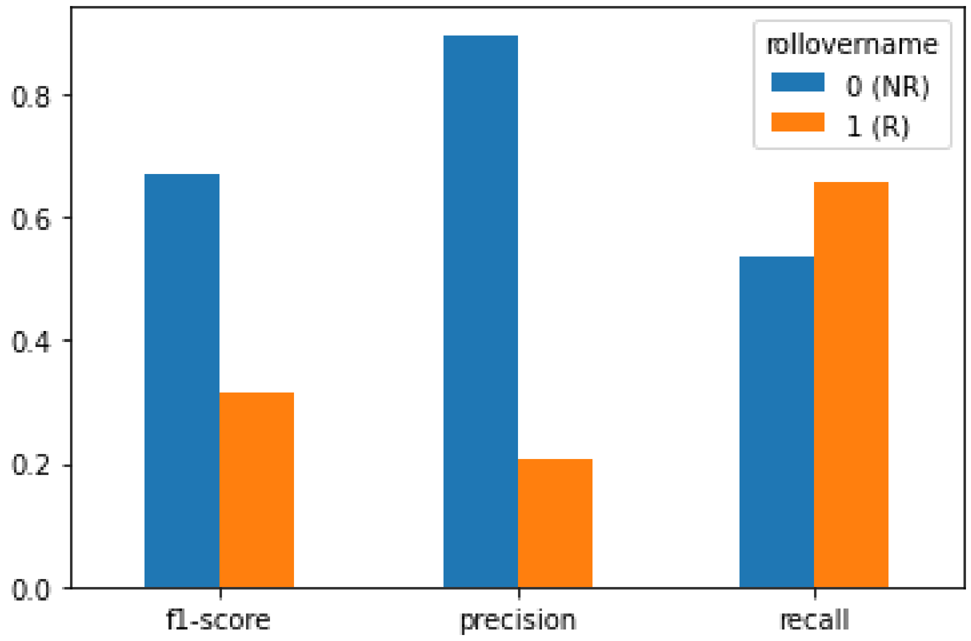
[22] Classification report for Random Forest model



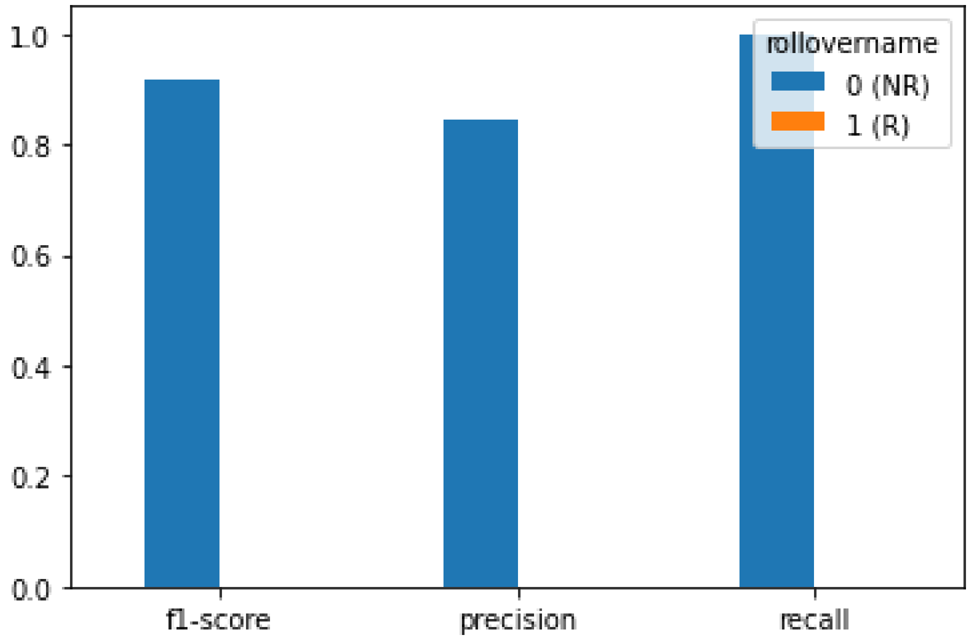
```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92     77111
           1       0.00      0.00      0.00     14127

    accuracy                           0.85     91238
   macro avg       0.42      0.50      0.46     91238
weighted avg       0.71      0.85      0.77     91238
```

[23] Classification report for Support Vector Machines model

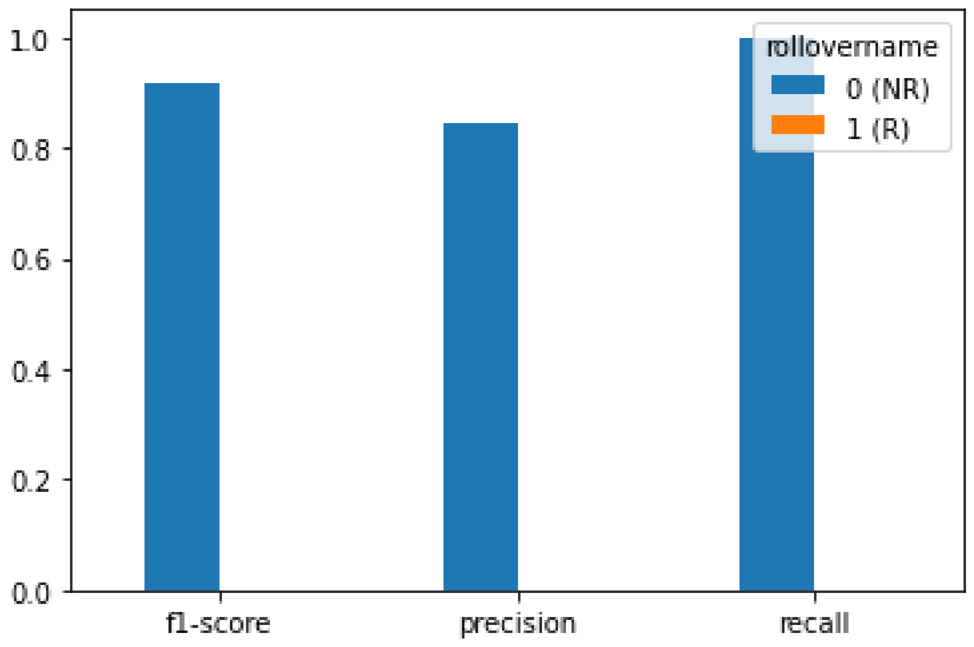
When the results are analyzed (in figures 24, 25, 26, 27), it is seen that the precision, recall, f1-score and accuracy values of complement naive bayes are different from the other 3 models. All 3 models (logistic regression, random forest and support vector machines) were trained more likely to predict no rollover, probably because there was an imbalance in the target feature. This explains why all performance metrics of rollover are 0 in charts. On the other hand, the reason why “complement naive bayes” rollover performance metrics are high is because this algorithm is robust against imbalanced datasets.



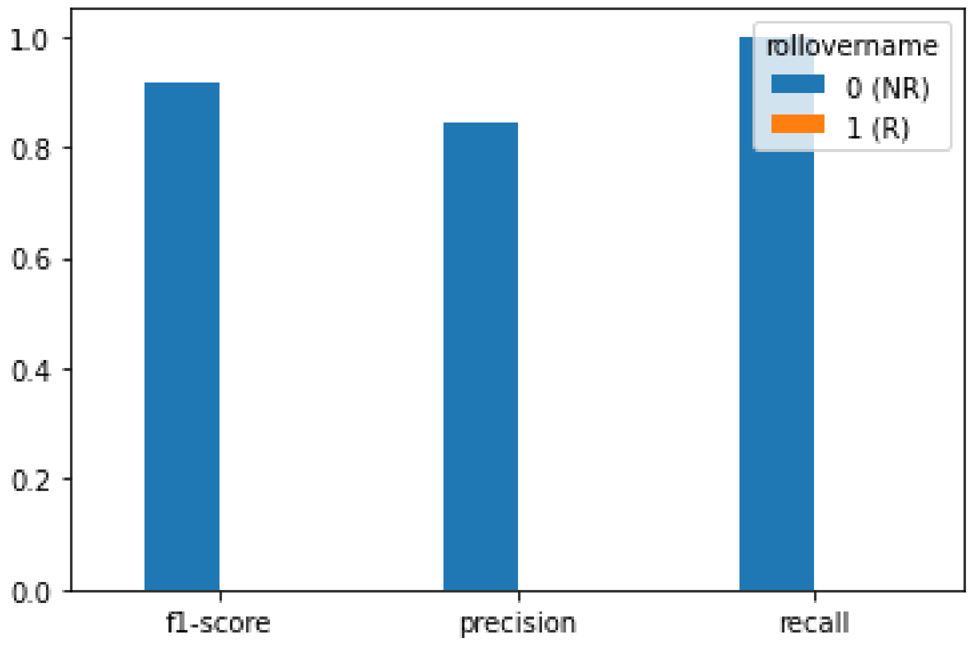
[24] Bar plot of f1-score, precision, and recall of Complement Naïve Bayes Model



[25] Bar plot of f1-score, precision, and recall of Logistic Regression Model

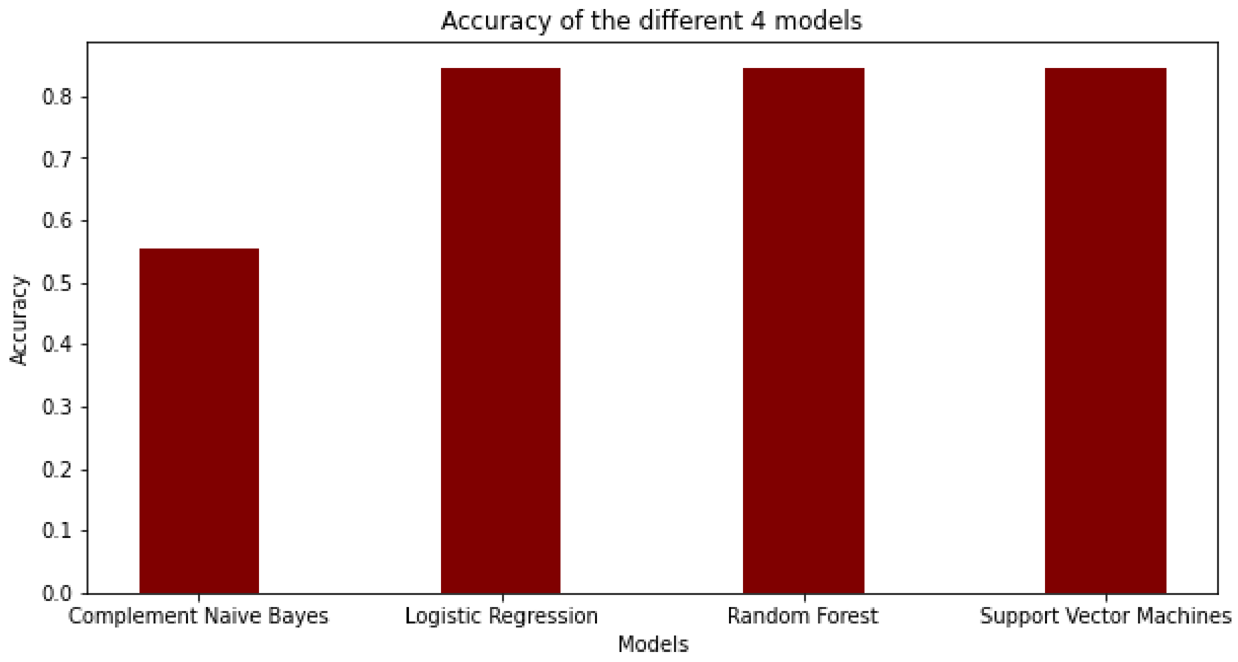


[26] Bar plot of f1-score, precision, and recall of Random Forest Model



[27] Bar plot of f1-score, precision, and recall of Support Vector Machines Model

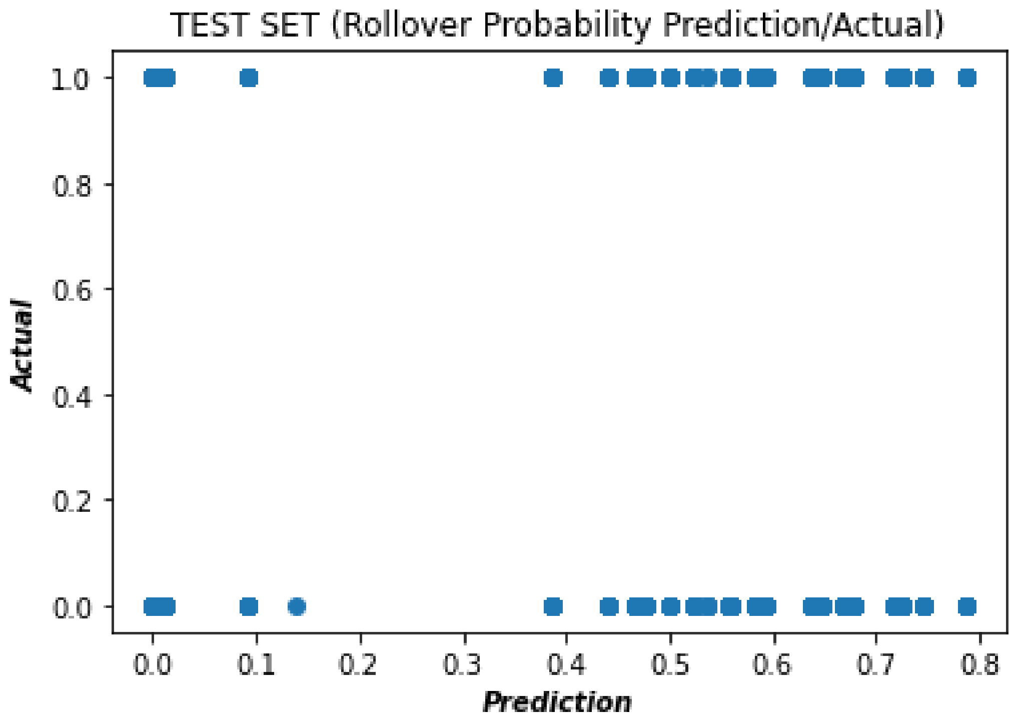
Accuracy values of 4 different models trained are shown in Figure 28. As can be seen, the accuracy of the complement naive bayes model is lower than the other three models. It is believed that complement naïve bayes model is not biased towards making prediction on no rollover value and this was the reason why it’s accuracy is smaller than the other models.



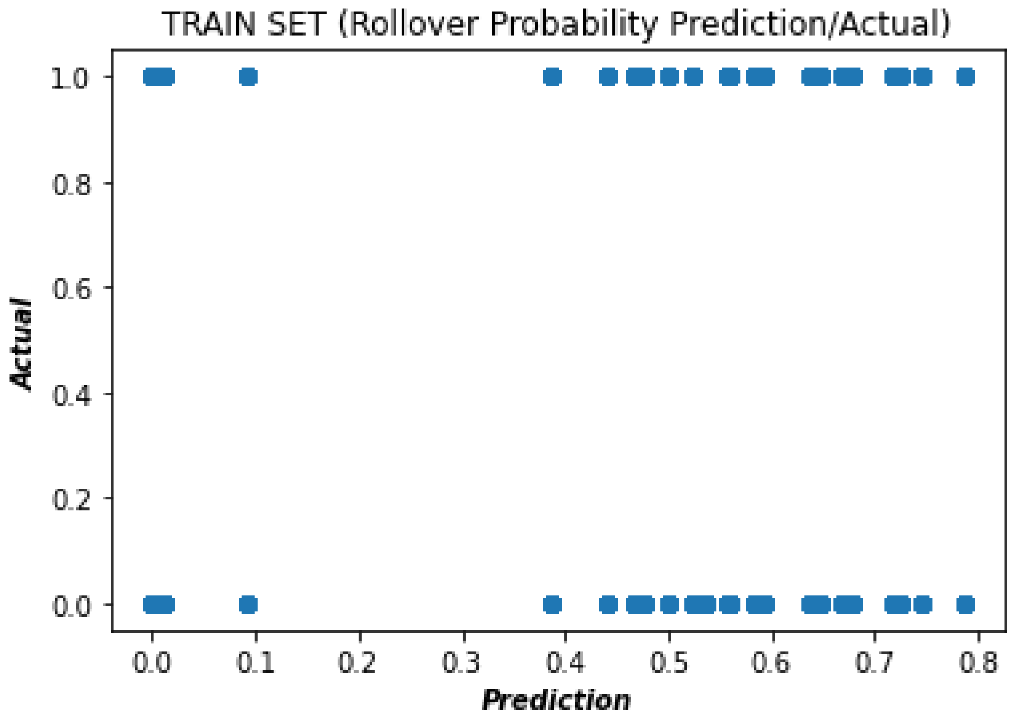
[28] Accuracy values of 4 different models

8. Interpretation

In this part, rollover and no rollover predictions were obtained for 4 different models trained using the *predict_proba()* function in the *sklearn* library. These predictions and actual values are then graphed. In order not to make the report too long, from the graphs drawn using test and train data, only those drawn for the naive bayes model will be shown here.



[29] Test set rollover probability prediction / actual for Complement Naïve Bayes model



[30] Train set rollover probability prediction / actual for Complement Naïve Bayes model

9. Appendix

Link to the data.csv file:
(https://drive.google.com/file/d/1ehgdsy_6ZzBK7u4gl04F6WX2CeZl99VG/view?usp=sharing)

10. References

[1] Gajawada, S. K. (2019, October 4). Chi-Square Test for Feature Selection in Machine learning. Retrieved from https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223