DATA ANYLSIS REPORT

TITLE:

Analysis on NHTSA FARS dataset to find out that in fatal crashes were the participant tests positive for marijuana the same as crashes were the participant tests positive for alcohol.

Author: Jayesh Fulzele

UIN: 668562417

Date: December 9, 2016

Table of contents: Page No

1. Introduction:

In 2014 National Highway Traffic Safety Administration (NHTSA) Fatality Analysis Reporting System (FARS) reported that around **32,675** people were killed in road accidents in that year[i]. Making it an average of 90 people per day. And around 5,492 people out of the total died because they were **under the influence of alcohol, drugs, or medication[ii].**

This report describes the findings of data analysis and implementation machine learning scheme and forming a predictive model on the FARS 2015 data set.

1.1 Purpose of the analysis:

The main goal of this analysis is to find out that in fatal crashes were the participant tests positive for marijuana the same as crashes where the participant tests positive for alcohol. Also, another objective is to build a machine learning model which can predict the result of if similar data was provided in the future. We will also see the distribution of fatal crashes on the all the cases in the dataset where alcohol and marijuana were involved.

1.2 Key Files:

1. Person.csv, This is the data file downloaded from the FARS website which consists information such as age, sex, state, alcohol and drug tests etc.
2. Project_EDA.ipynb, this the iPython notebook file where the Extrapolatory Data Analysis is performed on the person.csv file and the data is converted from numeric to nominal form using FARS User guide[iii] to make sense of the data.
3. Project_EDA-Numeric.ipynb, in this iPyhon notebook the data is further cleaned and condensed. It is then converted back to the numeric form so that Machine learning algorithm can be applied on it.
4. Project_RFC.ipynb, in this iPython notebook file the machine learning model is trained upon the training data and it is then used to predict the output of the test data. Further, here we measure the performance and accuracy of the model.
5. Project_analysis.ipynb, Project_analysis-actual-test and Project_analysis-actual total, these ipython notebook files are used to make distribution analysis of the data.

2. Methodology:

2.1 Extrapolatory Data Analysis:

In this section I used the Person.csv file downloaded from the FARS website and then I performed Extrapolatory Data Analysis on it. This file had numeric indicators of the nominal data in the fields so to understand what the information in this file actually meant. I used the FARS user guide for the year 2015[iii] and converted the data into its nominal form.

After this I examined the data to see which columns are necessary and which are not. I dropped some columns that I thought were unnecessary for our analysis as these columns gave least or no information regarding the presence of alcohol or use of marijuana in a person's system.

```
# After going through the FARS dataset User guige, I found the following columns to be least useful when forming relation with
# our requirement, Hence I've dropped these variables.
X = X.drop(['COUNTY','VE_FORMS','VEH_NO','MOD_YEAR','STR_VEH','FUNC_SYS','HARM_EV','MAN_COLL','SCH_BUS','IMPACT1','FIRE_EXP',
            'BODY_TYP','TOW_VEH','SEAT_POS','REST_USE','REST_MIS','AIR_BAG','HOUR','SPEC_USE','EMER_USE','MINUTE','MAKE',
            'MAK_MOD','ROLLOVER','EJECTION','EJ_PATH','EXTRICAT','DEATH_YR','DEATH_HR','DEATH_MN','DEATH_TM','LAG_HRS',
            'LAG_MINS','P_SF1','P_SF2','P_SF3','LOCATION', 'HOSPITAL'], axis=1)
```

Fig. Removing unnecessary data

I thinned down the data by grouping multiple fields together and replacing them with a single filed such as in the following figure:

```
In [96]: replacements = {
             'DRUG_DET': {'Behavioral|Drug Recognition Technician Determination':'Other'}
             }
         X.replace(replacements, regex=True, inplace=True)
```

Fig. Replacing certain data fields 1

```
#Cleaning Drug result columns as we are only intrested in Cannabinoid
replacements = {
    'DRUGRES1': {'Unknown if Tested': 'Not Tested for Drugs',
                 'Tested for Drugs, Results Unknown':'Not Reported',
                 'No Drugs Reported or Negative':'Reported Negative',
                 'Stimulant|Depressant|Narcotic|Hallucinogen|Phencyclidine|Inhalant|Anabolic Steroid|Tested for Drugs, Drugs Foun
    'DRUGRES2': {'Unknown if Tested': 'Not Tested for Drugs',
                 'Tested for Drugs, Results Unknown':'Not Reported',
                 'No Drugs Reported or Negative':'Reported Negative',
                 'Stimulant|Depressant|Narcotic|Hallucinogen|Phencyclidine|Inhalant|Anabolic Steroid|Tested for Drugs, Drugs Foun
    'DRUGRES3': {'Unknown if Tested': 'Not Tested for Drugs',
                 'Tested for Drugs, Results Unknown':'Not Reported',
                 'No Drugs Reported or Negative':'Reported Negative',
                 'Stimulant|Depressant|Narcotic|Hallucinogen|Phencyclidine|Inhalant|Anabolic Steroid|Tested for Drugs, Drugs Foun

}
X.replace(replacements, regex=True, inplace=True)
```

Fig. Replacing certain data fields 2

In the above filed I grouped all the other drugs apart from 'Cannabinoid' as 'Other Drugs' as our goal is to find if the person has used marijuana or not. I did similar process for the 'ALC_RES' columns where I grouped all the alcohol test results with BAC (blood alcohol count) greater that

0.08 as ‘Legally intoxicated’ as this is any reading above 0.07 is considered as Legally intoxicated by US Laws[iv]. And replace all values below 0.08 by 0.

```
None Given             51230
0.0                    21073
Legally intoxicated     8284
Name: ALC_RES, dtype: int64
```

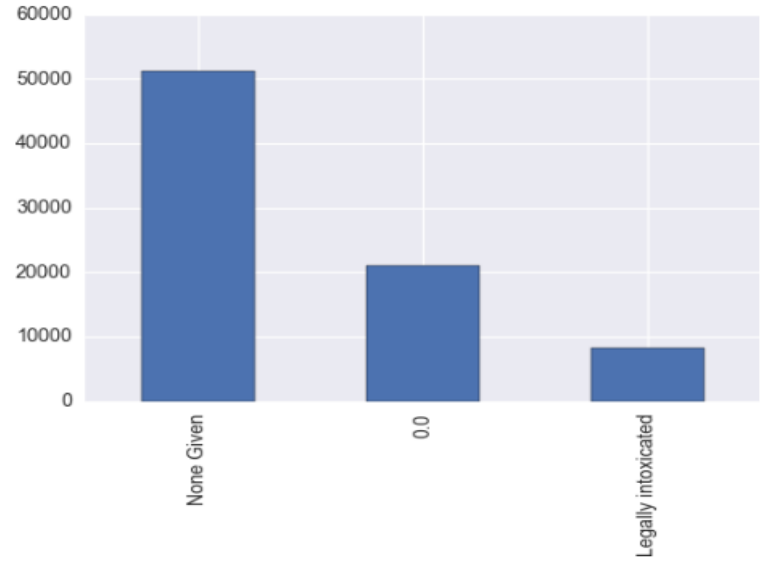
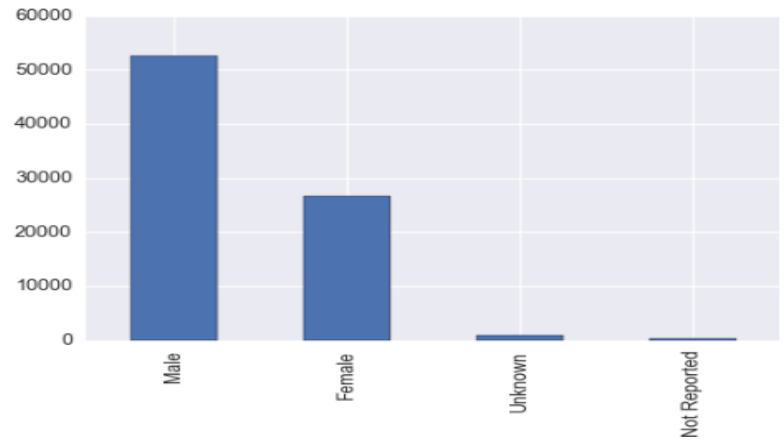


Fig. Distribution based on alcohol test results

I also replaced some of the data with minor or fields with ‘not reported’ or ‘unknown’ with majority fields.

```
Male            52598
Female          26631
Unknown           969
Not Reported      389
Name: SEX, dtype: int64
```




```
#Replacing Unknown and Not Reporthed sex with Male as this is the majority considering other lables
replacements = {
    'SEX': {'Unknown|Not Reported': 'Male',}
    }
X.replace(replacements, regex=True, inplace=True)
```

Fig. Replacing with majority fields

After cleaning all the data and fields I saved the data in a csv format to Project_EDA.csv file.

2.2 Converting the data back to numeric form:

After Performing the EDA and understanding the data, the next step is to build a machine learning algorithm that will predict any future test data. But before that it is required to convert the data into numeric form. For this, I again used the FARS User guide[iii] to convert the data into its numeric form.

For example:

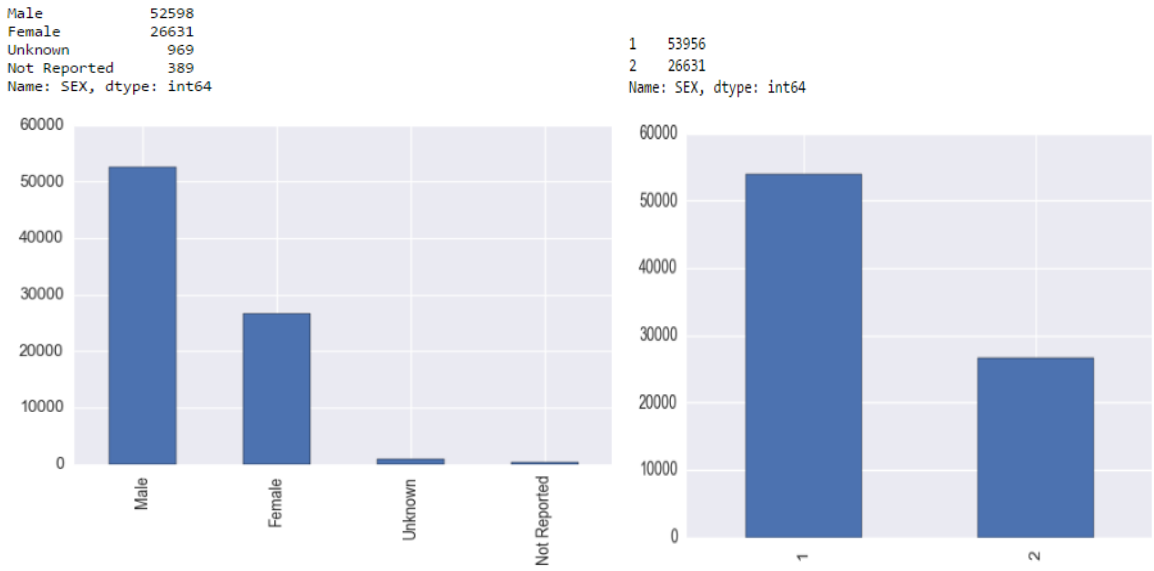


Fig. Nominal Form *Fig. Numeric Form*

In the above figure I replace the Unknown and not reported values with the majority label i.e. Male. And then used the user guide to convert into numeric by replacing Male with 1 and Female with 2.

After converting all the columns in the dataset, I added a new column to the data set named 'both' with two possible values 1 and 0. 1 specifying that the person involved in the fatal crash was under influence or had consumed alcohol and was tested positive for cannabinoid in at least one of the three 'DRUGRES' test and 0 if the afore said condition is not satisfied.

This gave me the number of people who had died in the crash while having consumed alcohol or having BAC above the legal limit and have used marijuana prior to crash.

I found 1,264 records out of 80,587 tested positive for the above case

```
#These are the cases where fatal crashes where the participant tests positive for marijuana the same as crashes
#where the participant tests positive for alcohol
X[X['both']==1]
```

80154	Wisconsin	550486	1	16	September	1	56.0	1	5	4	...	695	695	0	7
80160	Wisconsin	550489	1	17	December	2	28.0	2	1	4	...	695	0	0	0
80235	Wisconsin	550524	1	23	November	1	31.0	1	1	4	...	695	996	996	7
80319	Wyoming	560023	1	3	April	1	22.0	1	1	4	...	695	0	0	7
80496	Wyoming	560087	1	21	August	1	52.0	1	1	4	...	695	0	0	7
80504	Wyoming	560091	1	29	August	1	26.0	1	1	4	...	695	0	0	7
80515	Wyoming	560094	1	29	August	2	52.0	1	1	4	...	695	0	0	7
80532	Wyoming	560103	1	30	September	2	55.0	1	1	4	...	695	996	996	7
80544	Wyoming	560112	1	31	October	1	27.0	2	1	4	...	695	0	0	7
80559	Wyoming	560120	1	3	November	1	24.0	1	1	4	...	996	695	0	7

1264 rows × 26 columns

Fig. Number of people died in a crash while under influence of alcohol or having consumed alcohol and tested positive for marijuana usage.

After this I split the file into train and test set so that I could build a machine learning model to predict the future values. However, I dropped the 'both' column from the test data set so that this set could be used to predict future values.

2.3 Building the Machine Learning Model

In this section I use the training file created in the previous section to train Random Forrest classifier model. As this classifier uses only numeric values I will drop the categorical variables

```
#Dropping categorical vraibles
X = X.drop(['STATE','ST_CASE','MONTH'], axis=1)
```

Fig. dropping categorical variables

I stored the output of the training set, 'both' column, in to a variable 'y'. This is the feature that we are required to predict in the test set. Now I split the training set into train and test samples so that I could train my model as well as measure its performance.

2.3.1 Grid search:

First I start with a small number of tress and run the grid search. This, helps to find the best features suitable for this data set and this classifier.

```
#grid search to find best parameters for RFC
n_estimators = [50]
max_features = ['auto','log2']
min_samples_split = [1,2,3,4,5,6,7]
max_depth = [None, 10, 5]
criterion = ['entropy', 'gini']


rfc = RandomForestClassifier(n_jobs=3)
#Parameters of pipelines can be set using '__' separated parameter names:
estimator = GridSearchCV(rfc,
                         dict(n_estimators=n_estimators,
                              max_features=max_features,
                              min_samples_split=min_samples_split, max_depth=max_depth, criterion=criterion
                              ), cv=None, n_jobs=1)
```

Running the Grid search gives the following result for best features

```
# best features for the model
estimator.best_estimator_
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=6,
            min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=3,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

2.3.2 Training the model:

Now that the best features have been found I increase the number of trees to '2000' so that the Random Forest classifier gives the best prediction.

```
#Running the model with best features and with large number of trees
model = RandomForestClassifier(n_estimators=2000, max_depth=None, max_features='auto',
                               min_samples_split=6, criterion='entropy', n_jobs=3, oob_score=True)
model.fit(X, y)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=6,
            min_weight_fraction_leaf=0.0, n_estimators=2000, n_jobs=3,
            oob_score=True, random_state=None, verbose=0, warm_start=False)
```

I use the cross validation method to get the cross validation scores to measure the performance of the model.

```
from sklearn import cross_validation
scores = cross_validation.cross_val_score(model, X, y, cv=10)
```

```
scores
```

```
array([ 0.99851117,  0.99851117,  0.99884202,  0.99851117,  0.99884202,
        0.99950356,  0.99801423,  0.9991726 ,  0.9991726 ,  0.99867615])
```

```
mean_score = scores.mean()
std_dev = scores.std()
std_error = scores.std() / math.sqrt(scores.shape[0])
ci =  2.262 * std_error
lower_bound = mean_score - ci
upper_bound = mean_score + ci

print ("Score is %f +/-  %f" % (mean_score, ci))
```

```
Score is 0.998776 +/-  0.000291
```

The final score for this model is 0.998776 which is pretty decent and hence this model can be used to predict the outputs of the test set.

2.4 Predicting Test output:

I use the Test set generated in section 2.2 to predict the output. This set does not have 'both' column and hence is a representation of future unseen data to predict the class.

```
#predicting for y=1 in test set
y_hat = model.predict(T)
```

```
#preticted outputs
y_hat
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

After running the model on this test data I get an array of the predicted outputs. I will use this array to measure the accuracy of the predicted data and the actual data which was generated in section 2.2.

I use the confusion matrix to measure the accuracy of the predictions to the actual data.

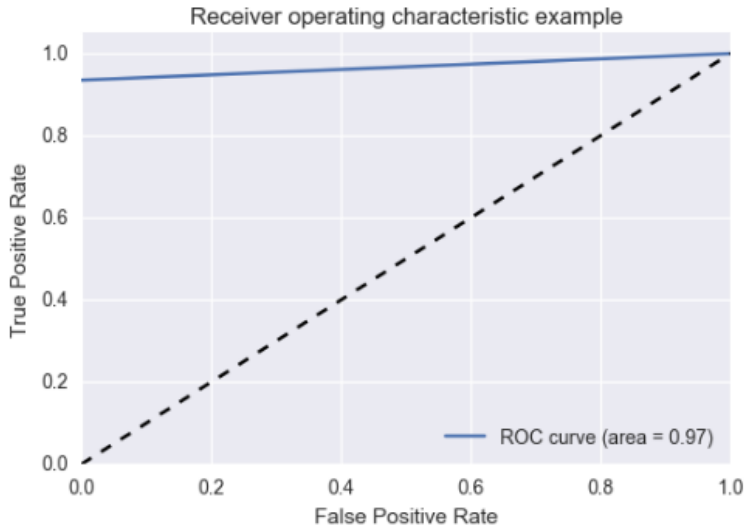
Confusion matrix

```
print (confusion_matrix(K, y_hat))
```

```
[[19838     0]
 [   20   289]]
```

Here, we see that out of 309 outputs the model correctly classified 289 instances and misclassified 20 instances. This gives an accuracy of 93.53%. Considering the performance constraints of my machine a lot of data was discarded so that I could run the model efficiently and still the model predicted 93.53% accurately is an indicator that for similar data this model can very likely predict if a person involved in a fatal crash had alcohol in his/her system and had used Marijuana prior to the crash.

I also found the AUC score to be 0.97 which is pretty decent. Further, bolstering the model prediction accuracy.



3. Results & Further Analysis:

3.1 Quality of Data:

The data that I downloaded from FARS website was numeric. I had numeric labels for its nominal equivalents. I had to convert these numeric indicators to nominal by referencing the FARS user guide[iii]. Then data was consistent but there were a lot of variables which had no relation with this analysis so I had to discard those columns. Also, using all of the data was not possible on my machine so thinning the data was the only option. Further, to implement the machine learning model I had to convert these back to numeric form.

3.2 Distribution Analysis:

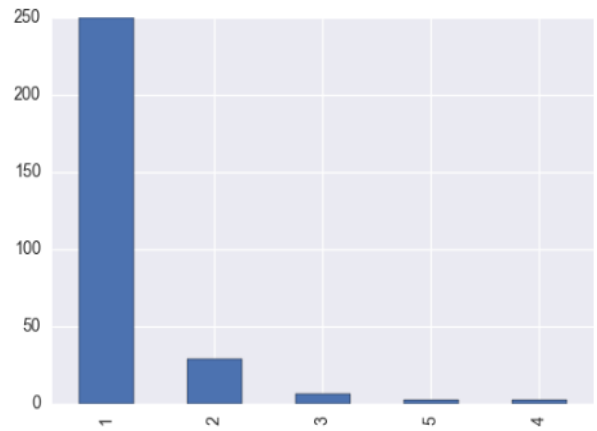
I use the iPython notebook files, Project_analysis.ipynb, Project_analysis-actual-test and Project_analysis-actual total, to perform analysis on the distribution of the data.

First I compare the distributions of predicted and actual test data.

(All of the following comparisons are for person involved in a fatal crash while having BAC above legal limit and tested positive for marijuana usage)

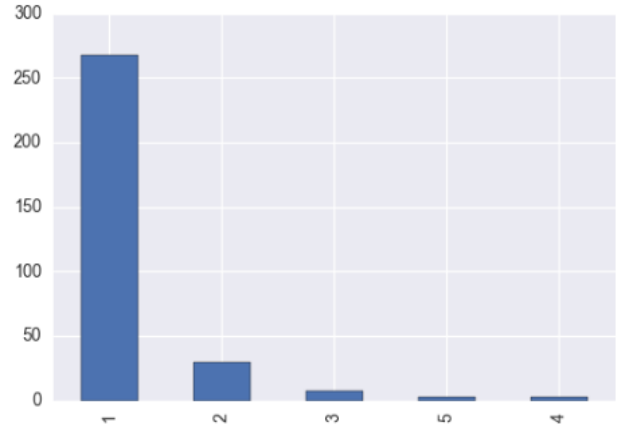
1. Comparing the number of people involved in the crash.

```
1    250
2     29
3      6
5      2
4      2
Name: PER_NO, dtype: int64
```



For predicted data

```
1    268
2     30
3      7
5      2
4      2
Name: PER_NO, dtype: int64
```

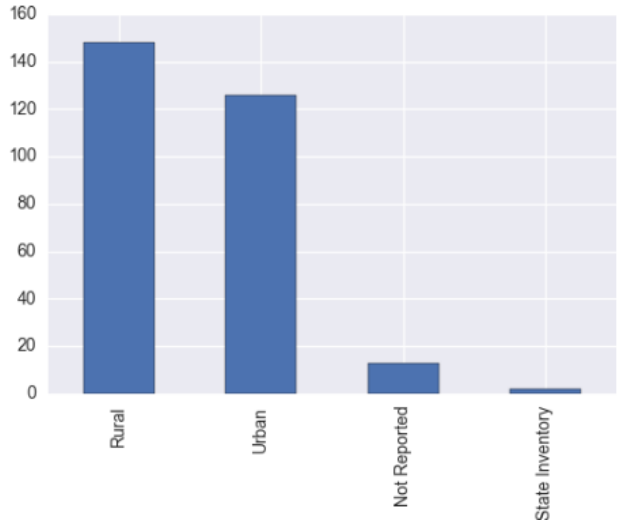


For actual data

We see that as per the actual data the number of people involved in a single crash, where number of people is 1, is 268 but in the predicted data it is 250. Further, where number of people is 2, the actual number is 30 and in predicted data it is 29. Where the number of people involved is 3 the actual is 7 and the predicted is 6. For other numbers the predicted and actual values are same.

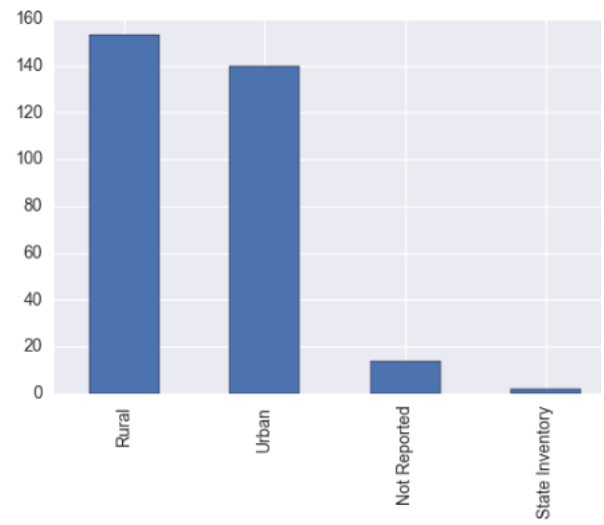
2. Comparing Rural and Urban crashes

```
Rural                                148
Urban                                126
Not Reported                          13
Trafficway Not in State Inventory      2
Name: RUR_URB, dtype: int64
```



For predicted data

```
Rural                                153
Urban                                140
Not Reported                          14
Trafficway Not in State Inventory      2
Name: RUR_URB, dtype: int64
```

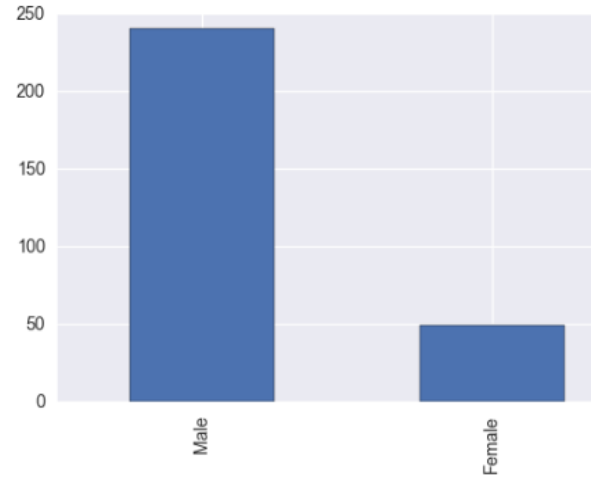


For actual data

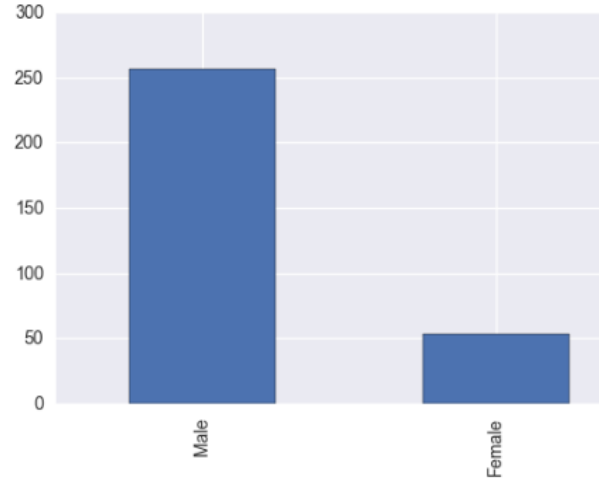
We see that as per the actual data the number of people belonging to a rural area that are involved in a crash is 153 but in the predicted data it is 148. For urban area the predicted value is 126 but the actual value is 140. For the not reported cases the actual is 14 and the predicted number of people is 13. And it's the same for the remaining.

3. Comparing gender

```
Male      240
Female     49
Name: SEX, dtype: int64
```



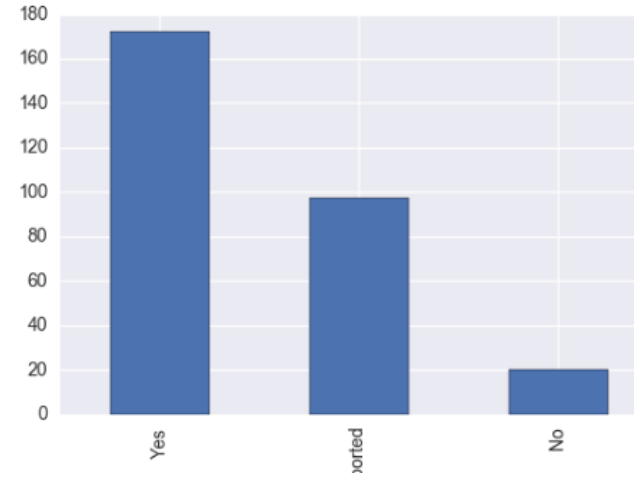

```
Male      256
Female     53
Name: SEX, dtype: int64
```



We see that as per the actual data the number of males is 256 but the predicted is 240. The actual number of Females is 53 and the predicted 49.

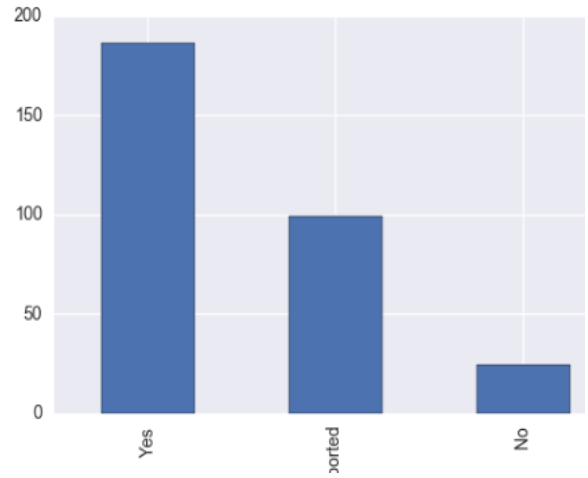
4. Comparing if the person had an alcoholic drink or not.

```
Yes             172
Not Reported     97
No               20
Name: DRINKING, dtype: int64
```



For predicted data

```
Yes             186
Not Reported     99
No               24
Name: DRINKING, dtype: int64
```

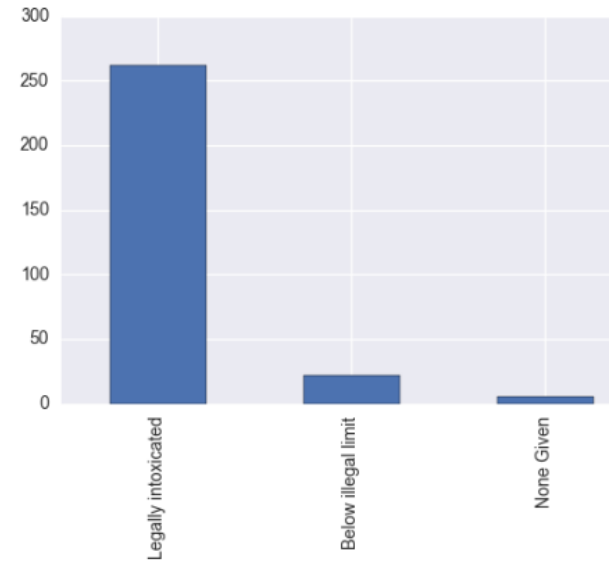


For actual data

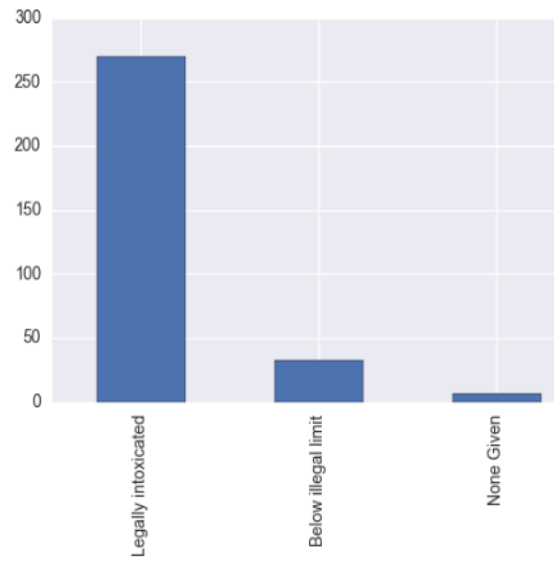
The predicted value for the people who had a drink is 172 but the actual value is 186. For the not reported cases the predicted value is 97 and the actual value is 99. For the cases that reported No. the predicted value is 20 but the actual value is 24.

5. Comparing Alcohol test result

```
Legally intoxicated    262
Below illegal limit     22
None Given               5
Name: ALC_RES, dtype: int64
```

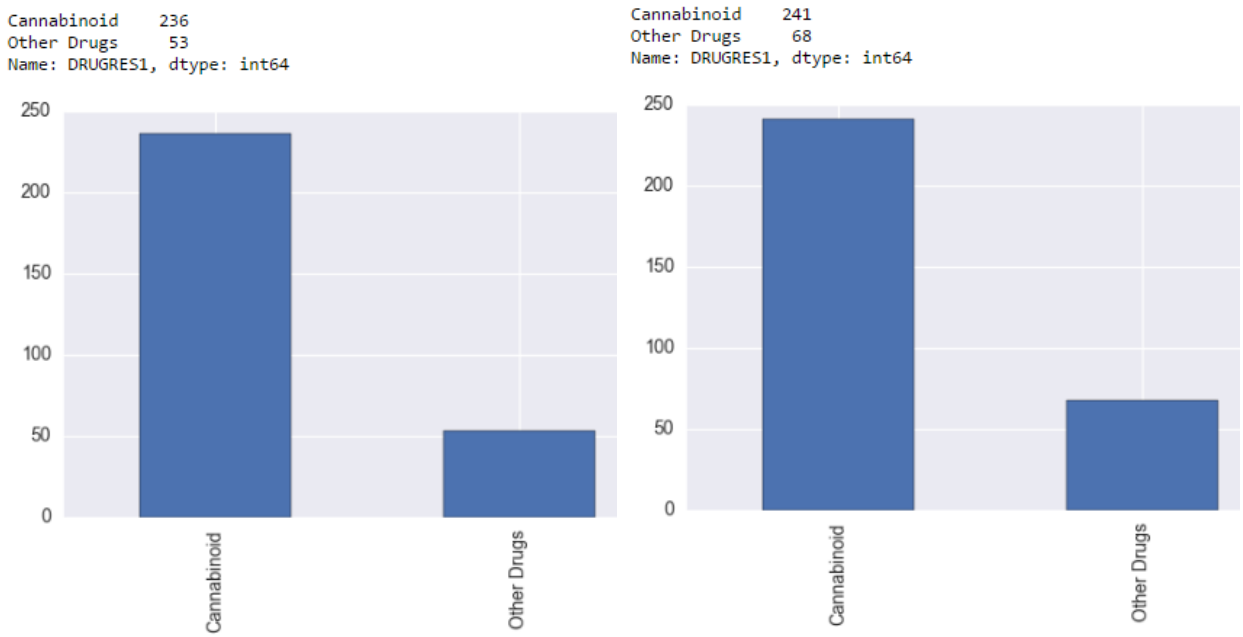



```
Legally intoxicated    270
Below illegal limit     33
None Given               6
Name: ALC_RES, dtype: int64
```



The predicted value for the people who were legally intoxicated is 262 but the actual value is 270. For the below illegal limit the predicted value is 22 and the actual value is 33. For the cases that had not given the test the predicted value is 5 and the actual is 6.

6. Comparing distribution based on Drug result 1.



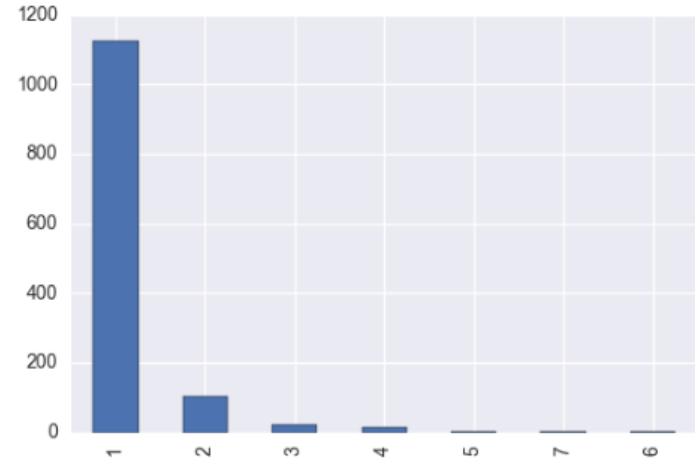
The Predicted value for the people who tested positive for cannabinoid is 236 and the actual value is 241. The predicted value for other drugs is 53 and the actual value is 68.

We see that the results are generally comparable with each other with some error.

Analysis on the whole data

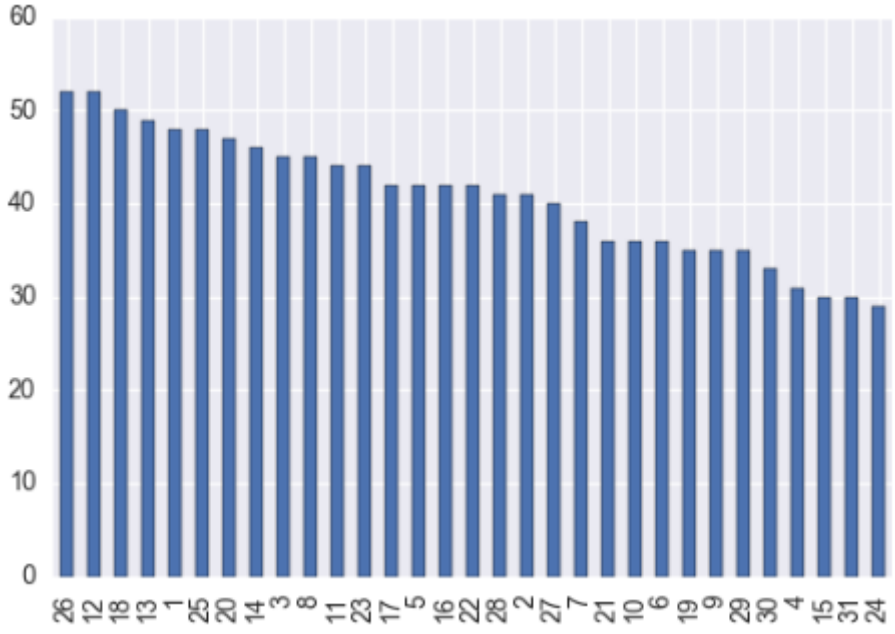
1. Distribution of Number of people involved in a single crash

```
1    1123
2     101
3      23
4      13
5       2
7       1
6       1
Name: PER_NO, dtype: int64
```



We see that the majority of people involved in the crash were travelling alone in their vehicle before the crash.

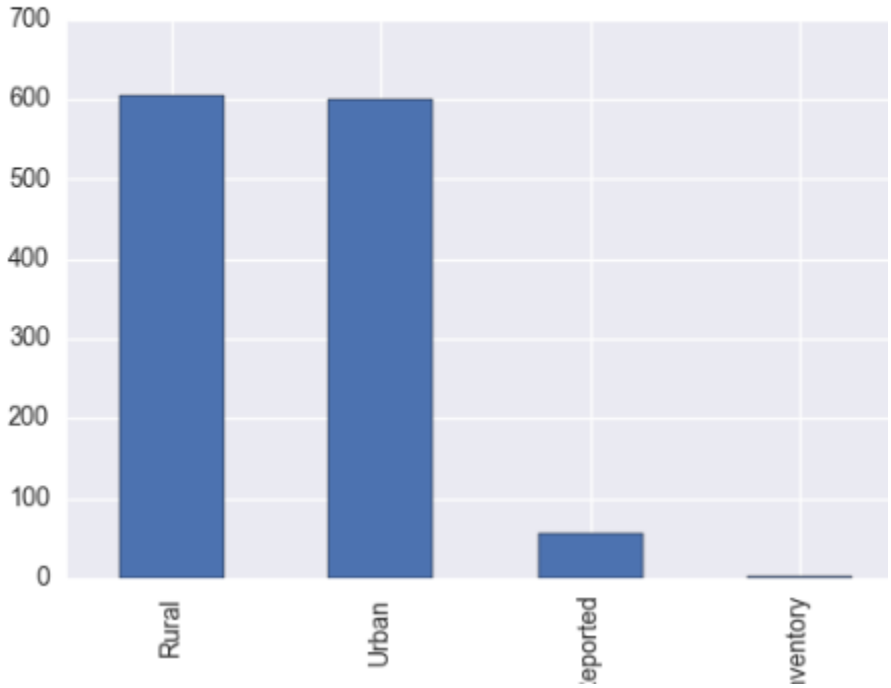
2. Distribution of the frequency of days in a month where people were involved in crashes.



We see that the most frequent days were 26th and 12th day of a month.

3. Rural and Urban case distribution

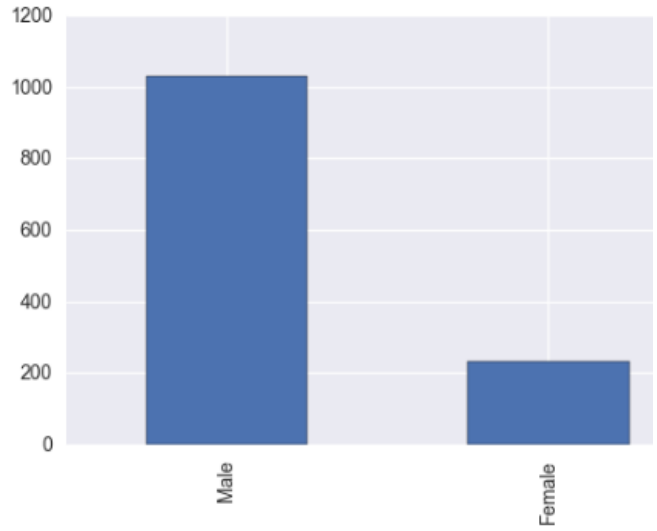
```
Rural                                605
Urban                                601
Not Reported                          56
Trafficway Not in State Inventory      2
Name: RUR_URB, dtype: int64
```



It is seen that the number of crash cases is almost identical in both urban and rural areas.

4. Distribution of Gender

```
Male      1030
Female     234
Name: SEX, dtype: int64
```



We see that the likelihood of a person involved in a fatal crash is a male is more than that of being a female.

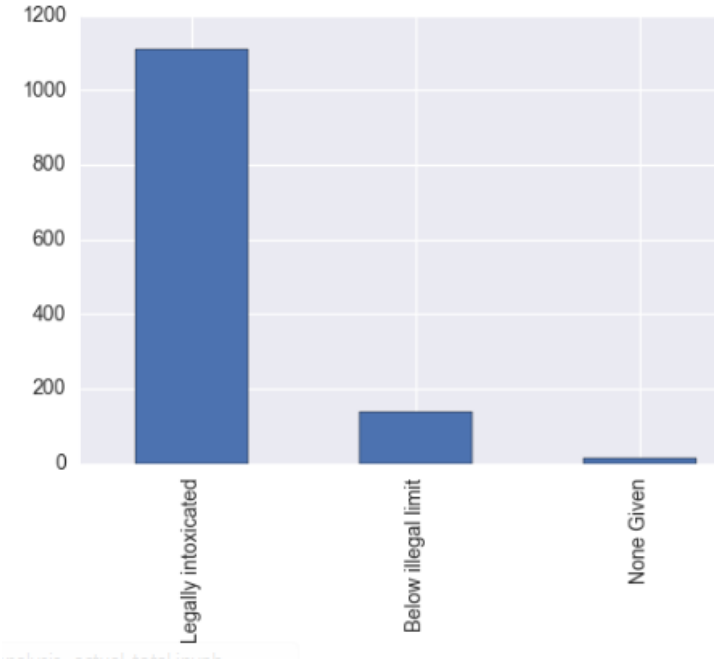
5. Distribution of Age

21.0	74
23.0	67
22.0	64
25.0	64
24.0	52
26.0	51
20.0	43
29.0	42
19.0	39
28.0	39
32.0	39
34.0	35
27.0	35
31.0	33
30.0	32
33.0	32

We find that the most people involved in crashes were in their 20's.

6. Distribution of Alcohol test result

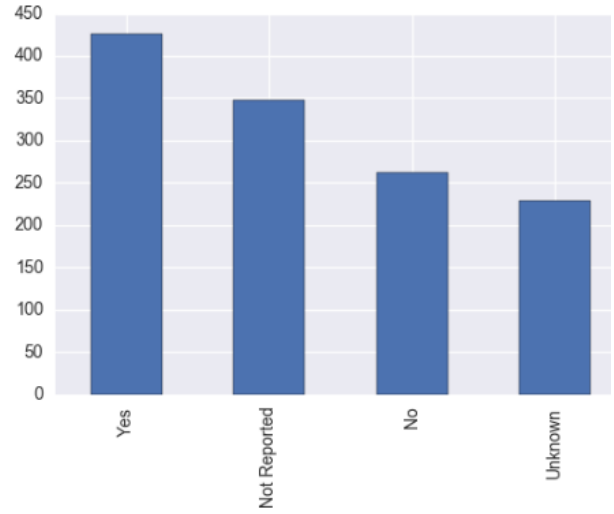
```
Legally intoxicated    1110
Below illegal limit     138
None Given               16
Name: ALC_RES, dtype: int64
```



We see that of the total number of people involved in a crash majority had alcohol above the legal limit in their blood.

7. Drug reported

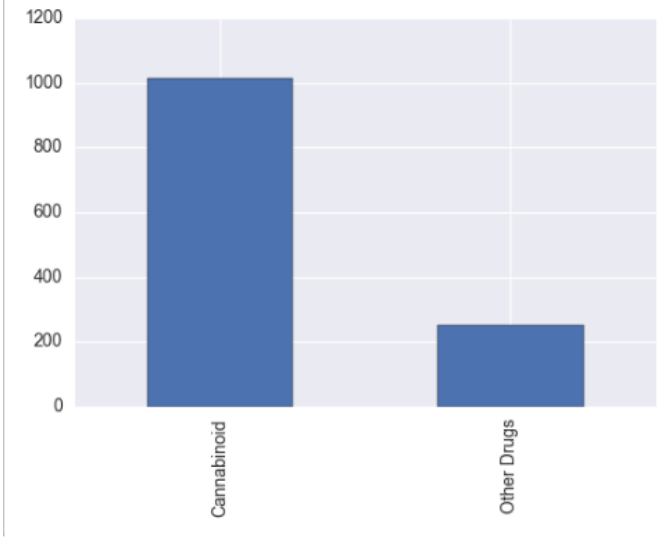
```
Yes             426
Not Reported    347
No              262
Unknown         229
Name: DRUGS, dtype: int64
```



This is an ambiguous column. The Unknown label is the ambiguous factor over here since it can be any one of the other cases. And since there are 229 people who may or may not have taken the drug test this might give rise to some problem.

8. Drug test result

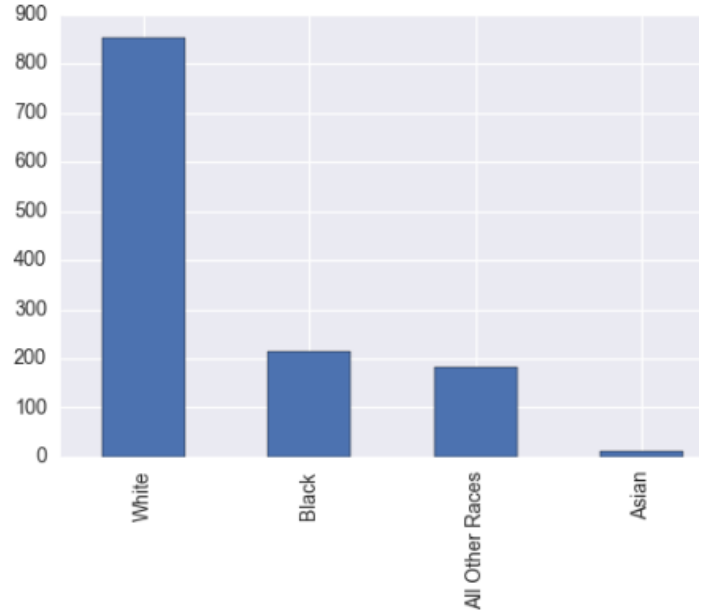
```
Cannabinoid    1012
Other Drugs     252
Name: DRUGRES1, dtype: int64
```



We see that of the people involved in crashes 1012 had Cannabinoid in their system while 252 people had some other drug abuse. This may be one of the factors resulting in a fatal crash.

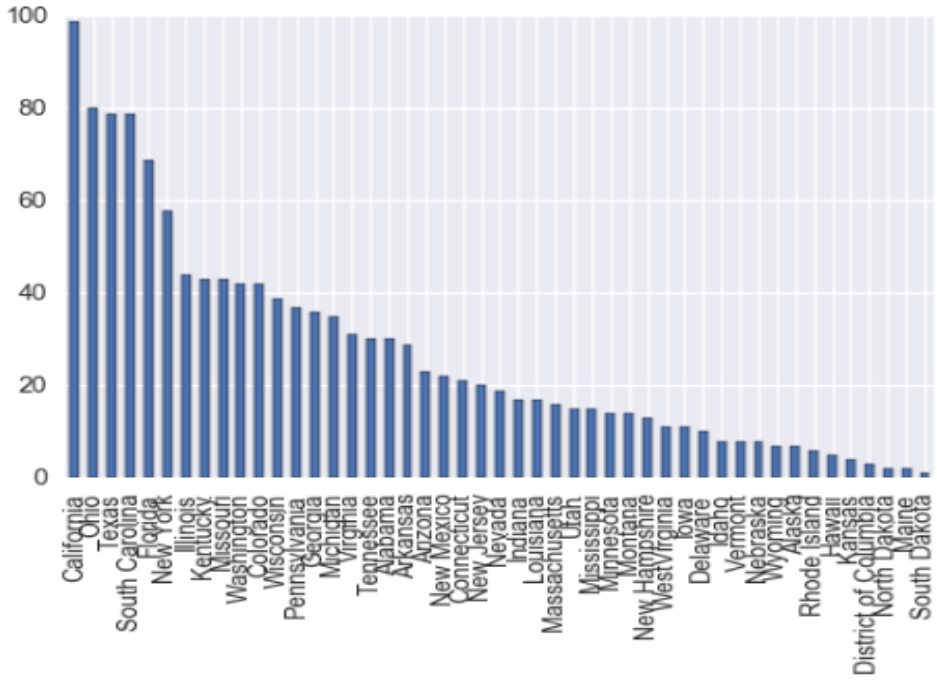
9. Distribution with respect to Race of the person

```
White              854
Black              216
All Other Races    182
Asian               12
Name: RACE, dtype: int64
```



Of all the crashes involving alcohol and marijuana the majority of people involved in crash were White whereas 216 were Black and the rest were Asian and other races.

10. Distribution with respect to the Sate to which a person in the crash belonged to.



We see that California was the state which reported highest crashes involving marijuana and alcohol numbering to 99 whereas, South Dakota was the state with least number of similar crashes numbering to 1.

Form the above analysis it is safe to say that the probability of a person involved on a fatal crash involving alcohol and marijuana use is likely to be a White male in his 20's.

4. Conclusion:

We have analyzed the data and found the distribution of people involved in a fatal crash involving alcohol and marijuana. We have successfully built a model which can predict that whether a person involved in a fatal crash had alcohol and marijuana in his test results, if similar data is provided, with a good accuracy.

The accuracy might be suffered since a lot of data was discarded to provided computational efficiency since my machine was not capable of handling this large amount of data. The result might have been better if I had used distributed systems to analyze and train the data.

However, in conclusion we can say that we have successfully analyzed the data and successfully used a machine learning scheme on the data to build a predictive model which can predict the outputs of similar unseen data provided in future.

5. Reference:

i. http://www-fars.nhtsa.dot.gov/Main/index.aspx
ii. http://www-fars.nhtsa.dot.gov/People/PeopleDrivers.aspx
iii. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812315 - FARS user guide
iv. http://alcoholpolicy.niaaa.nih.gov/Blood_Alcohol_Concentration_Limits_Adult_Operators_of_Noncommercial_Motor_Vehicles.html