CA02-Machine Learning

B9DA104

Bharat Jethwani Student Number- 10519364 [Course title]

Contents

Ch.1.

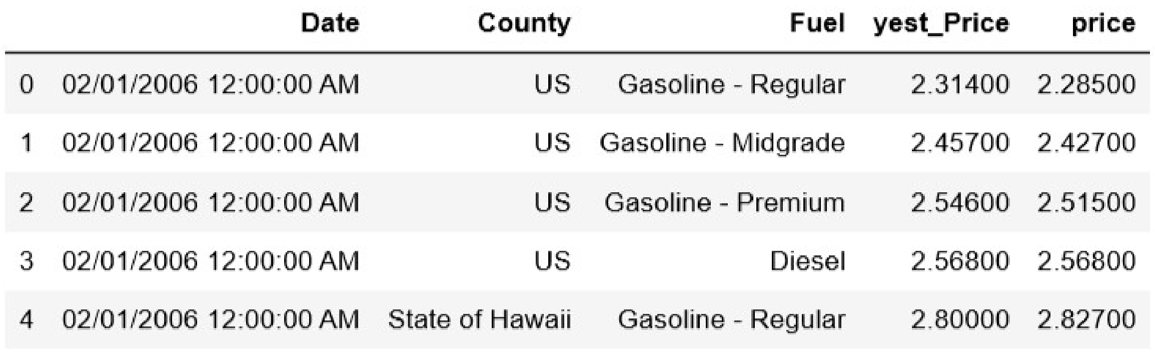
1.1 Business and Data Understanding:

Regression Project:

The dataset for this project was taken from "https://catalog.data.gov/dataset/aaa-fuel-prices-52bf0" consists of Gasoline Prices for different County's from 01-01-2006 from 07-01-2012 by Fuel-type. The primary objective of this study was to forecast gasoline prices using Linear Regression. Other purposes of this research are as follows:

- Compare the performance of Linear Regression on the original dataset(Consisting of Fuel type and newly created variable Yesterdays's price) with a subset of the original dataset(not containing Fuel-type and Yesterday's price)
- Comparing the performance of different Regression models on the original dataset.

Below fig displays the head values for the Regression project.



	Date	County	Fuel	yest_Price	price
0	02/01/2006 12:00:00 AM	US	Gasoline - Regular	2.31400	2.28500
1	02/01/2006 12:00:00 AM	US	Gasoline - Midgrade	2.45700	2.42700
2	02/01/2006 12:00:00 AM	US	Gasoline - Premium	2.54600	2.51500
3	02/01/2006 12:00:00 AM	US	Diesel	2.56800	2.56800
4	02/01/2006 12:00:00 AM	State of Hawaii	Gasoline - Regular	2.80000	2.82700

Fig1. Attributes for Regression project

Data Understanding:

Below, visualizations compare different attributes with each other and themselves.

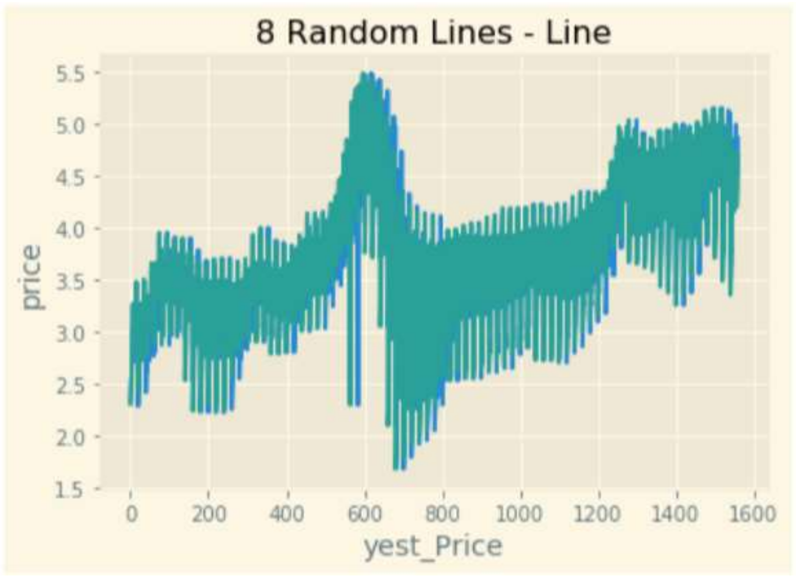


Figure1. Yester price vs. Todays Price

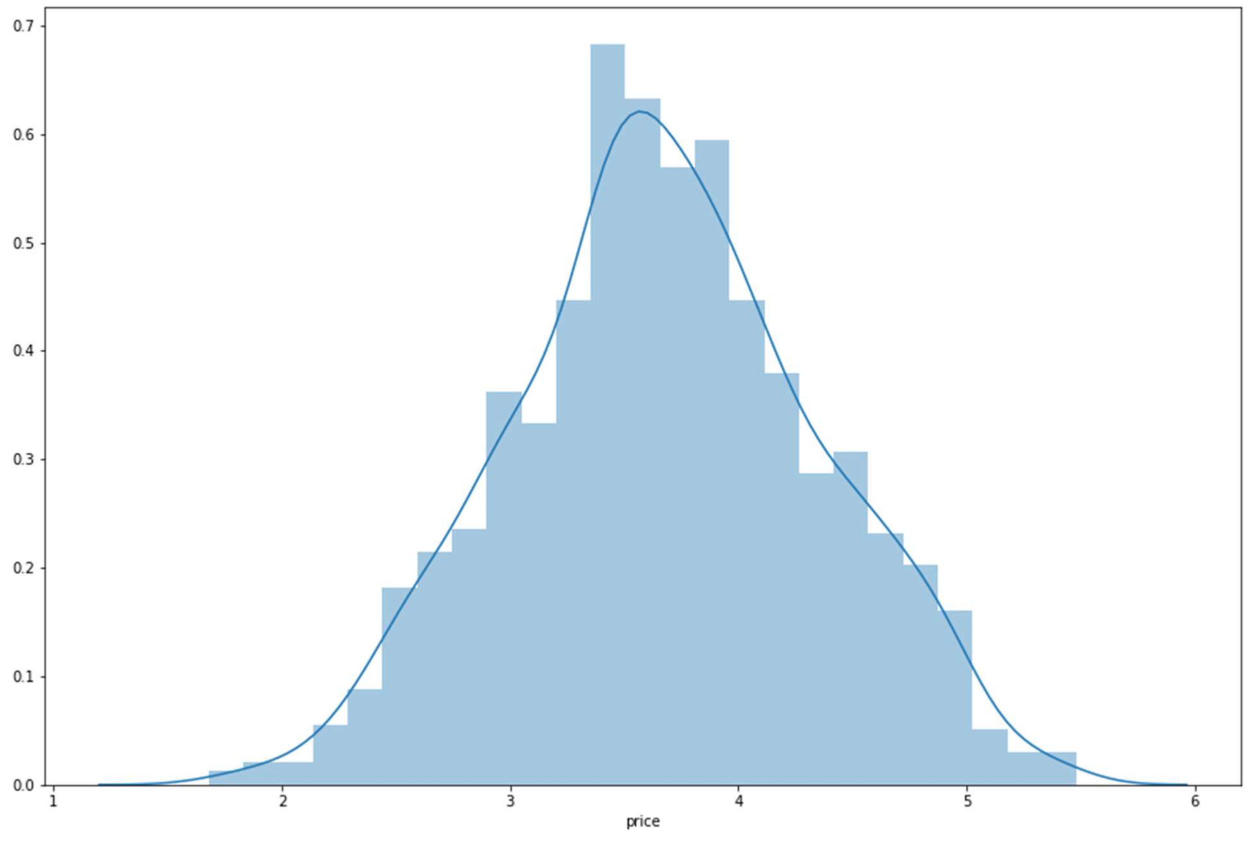


Figure2. Price Distribution

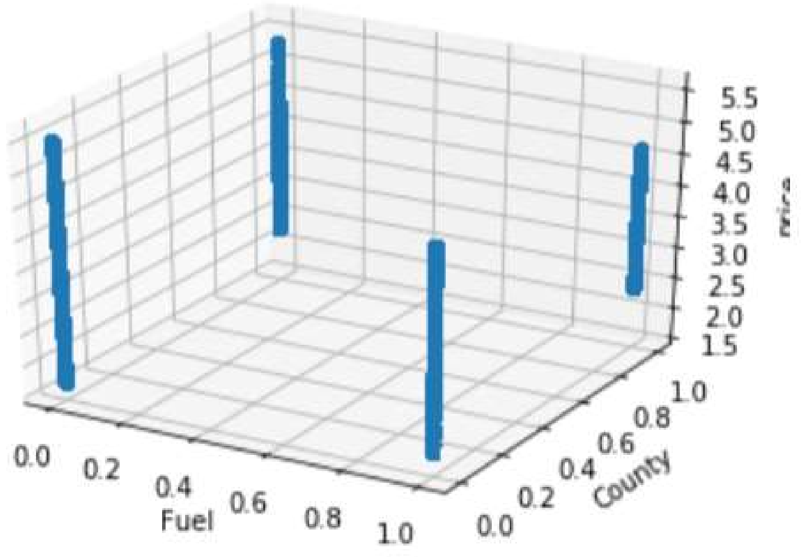
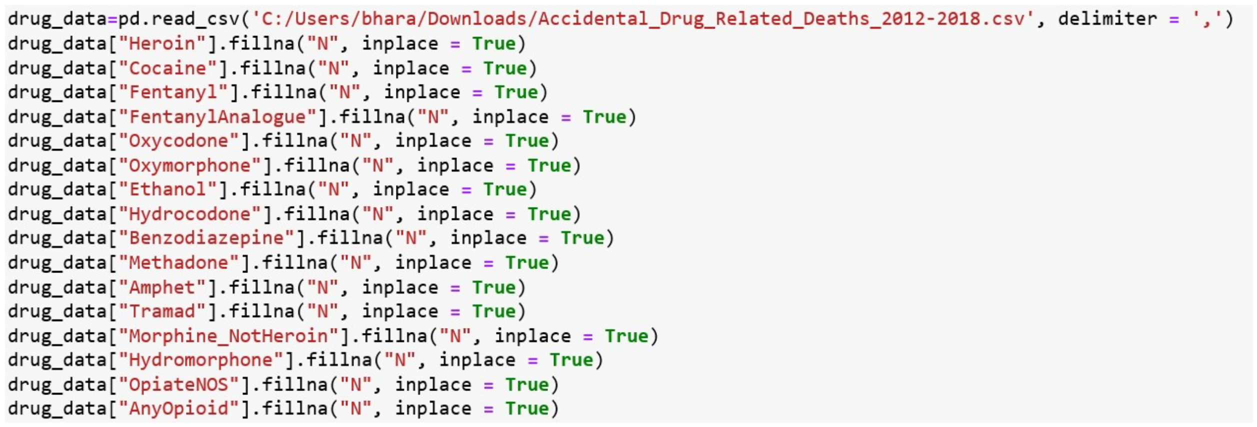


Figure3. Price distribution over County by Fuel category

Classification Project:

The dataset for this project was taken from "https://catalog.data.gov/dataset/accidental-drug-related-deaths-january-2012-sept-2015"; it contains the Summary of all accidental substance overdose deaths from 2012 to 2018. A value of "Y" below the columns indicates the presence of a specific material and "" (blank) values if not present. Blank values were replaced in data pre-processing as shown in Fig1.



```
drug_data=pd.read_csv('C:/Users/bhara/Downloads/Accidental_Drug_Related_Deaths_2012-2018.csv', delimiter = ',')
drug_data["Heroin"].fillna("N", inplace = True)
drug_data["Cocaine"].fillna("N", inplace = True)
drug_data["Fentanyl"].fillna("N", inplace = True)
drug_data["FentanylAnalogue"].fillna("N", inplace = True)
drug_data["Oxycodone"].fillna("N", inplace = True)
drug_data["Oxymorphone"].fillna("N", inplace = True)
drug_data["Ethanol"].fillna("N", inplace = True)
drug_data["Hydrocodone"].fillna("N", inplace = True)
drug_data["Benzodiazepine"].fillna("N", inplace = True)
drug_data["Methadone"].fillna("N", inplace = True)
drug_data["Amphet"].fillna("N", inplace = True)
drug_data["Tramad"].fillna("N", inplace = True)
drug_data["Morphine_NotHeroin"].fillna("N", inplace = True)
drug_data["Hydromorphone"].fillna("N", inplace = True)
drug_data["OpiateNOS"].fillna("N", inplace = True)
drug_data["AnyOpioid"].fillna("N", inplace = True)
```

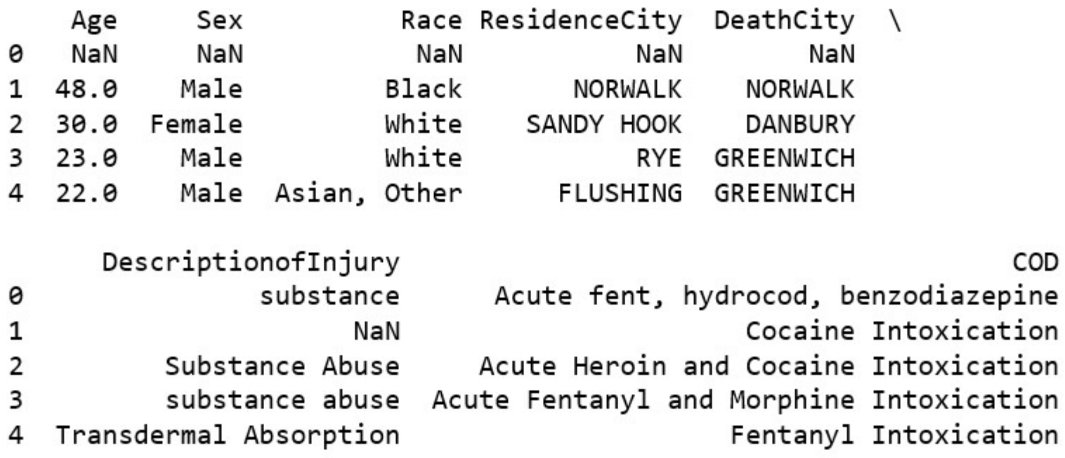
Fig2. Replacing blank values with N for all drug named attributes

The aim of this study is "Comparative analysis of Machine Learning Classifiers on Accidental related deaths dataset." For doing the same five Classifiers listed below were used for binomial classification to detect the presence of the following drugs :"Heroin", "Cocaine", "FentanylAnalogue", "Oxycodone", "Ethanol", "Hydrocodone", "Benzodiazepine", "Methadone", "Amphet", "Tramad", "Hydromorphone", "OpiateNOS", "AnyOpioid".

Classifiers Used:

- Logistic Regression
- Decision Tree
- Naive Bayes
- SVM
- KNN

The below figure displays the head values for the selected attribute.



```
    Age     Sex          Race ResidenceCity  DeathCity  \
0   NaN     NaN           NaN           NaN        NaN
1  48.0    Male         Black       NORWALK    NORWALK
2  30.0  Female         White    SANDY HOOK    DANBURY
3  23.0    Male         White           RYE  GREENWICH
4  22.0    Male  Asian, Other      FLUSHING  GREENWICH

     DescriptionofInjury                                 COD
0              substance      Acute fent, hydrocod, benzodiazepine
1                    NaN                   Cocaine Intoxication
2        Substance Abuse     Acute Heroin and Cocaine Intoxication
3        substance abuse  Acute Fentanyl and Morphine Intoxication
4 Transdermal Absorption                   Fentanyl Intoxication
```

Fig 3. Attributes used for Classification Project

Data Understanding

Below, visualizations compare different attributes with each other and themselves.

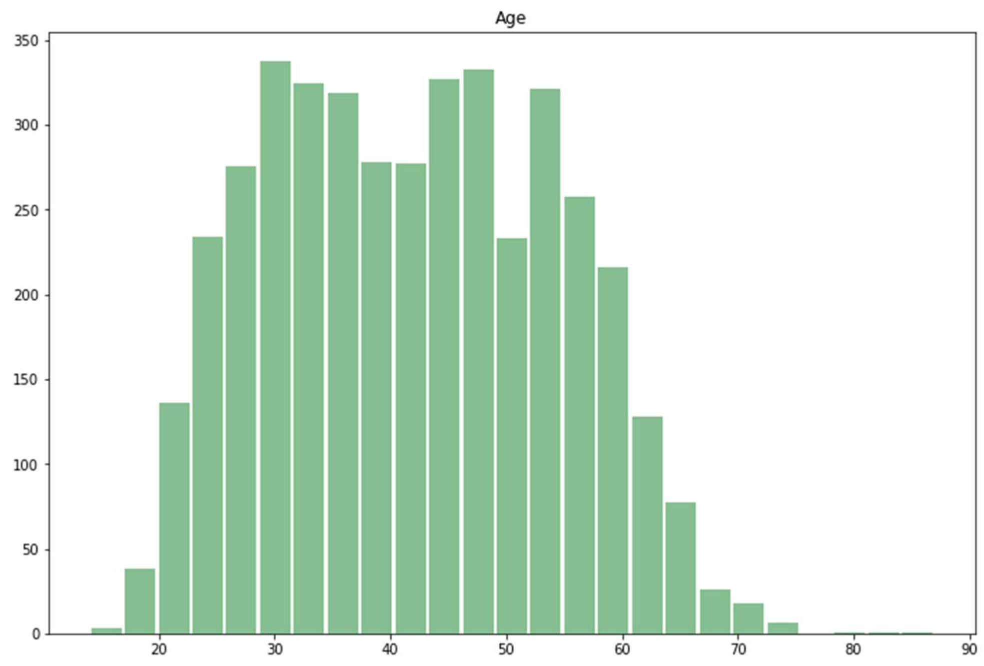


Figure1. Age Distribution

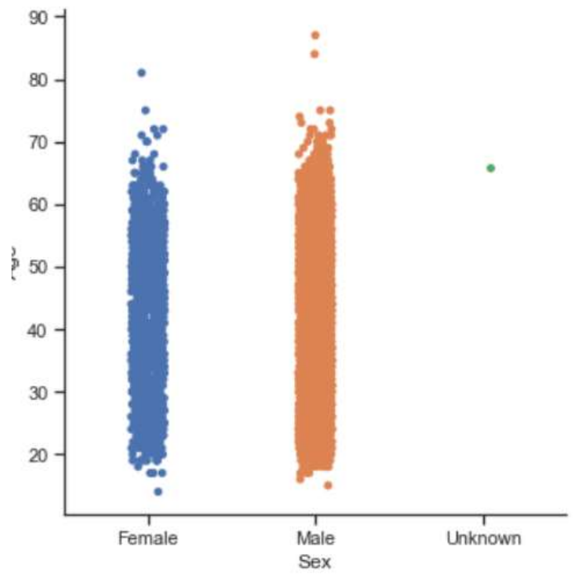


Figure2. Comparing Sex distribution over Age.

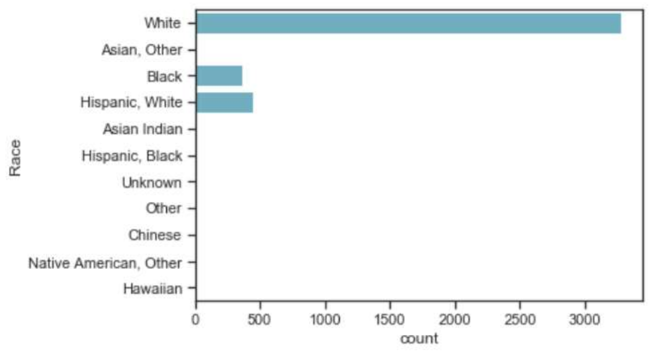


Figure3. Race Distribution

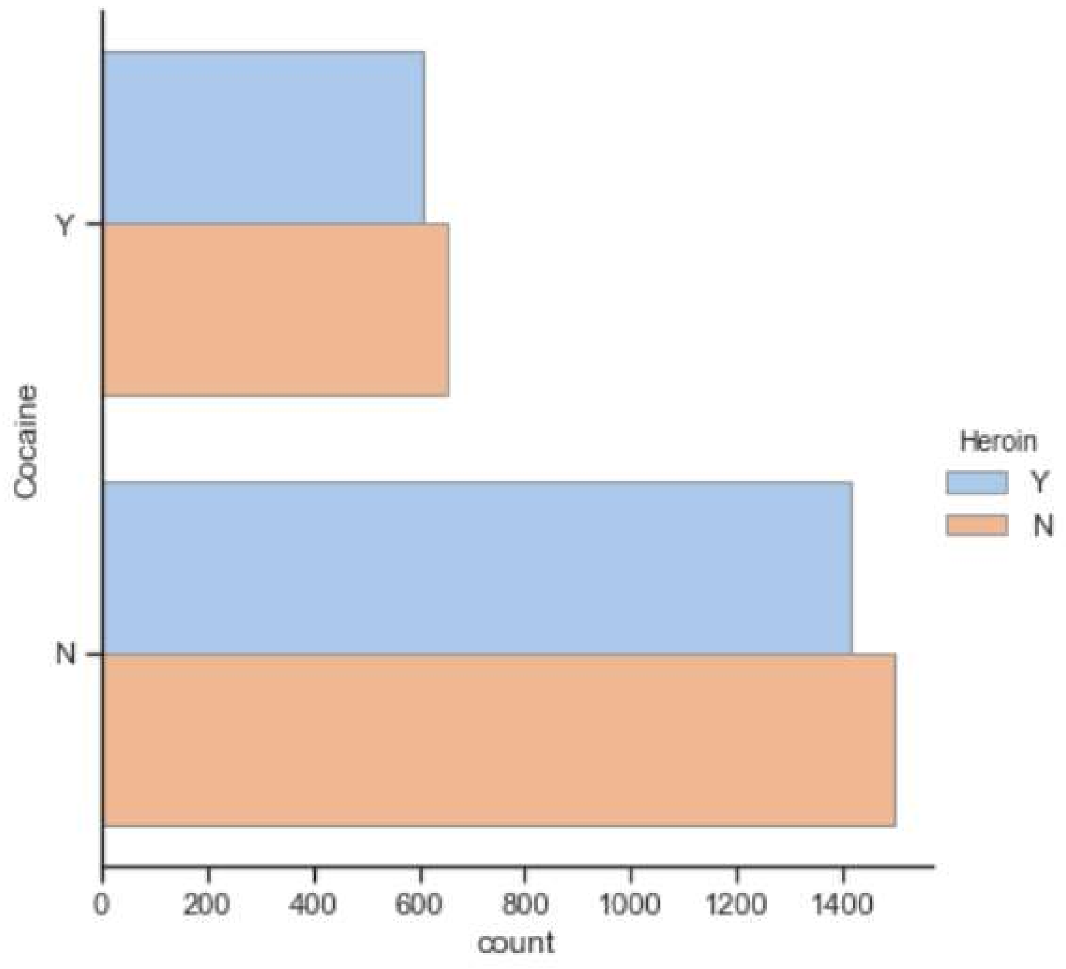


Figure4. Count analysis of Cocaine injected along with injected heroin

Ch.2.

2.1 What is Data Pre-Processing

Data scientists spend about 60% of their project time on cleaning and transforming data, followed by 19% of project time in collecting data, i.e., 80% of project time is spent on data pre-processing. (Press, 2016)

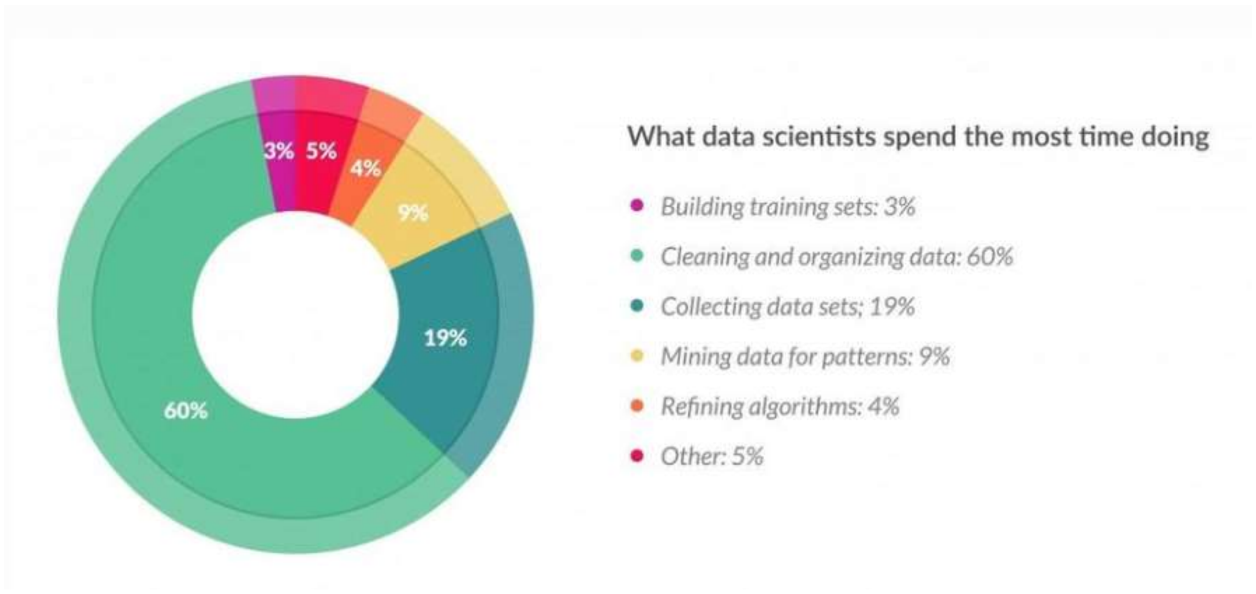


Fig4. Time Spent on various tasks (Press, 2016)

Data pre-processing is a data mining technique involving raw and unstructured data processing in a comprehensible format. (Techopedia.com, 2020) It is essential to use correct modeling techniques for selecting clean sub-sets or replacing missing data. During this process, the required data is also integrated, and several records incorporated.

Steps Involved in Data Preprocessing:

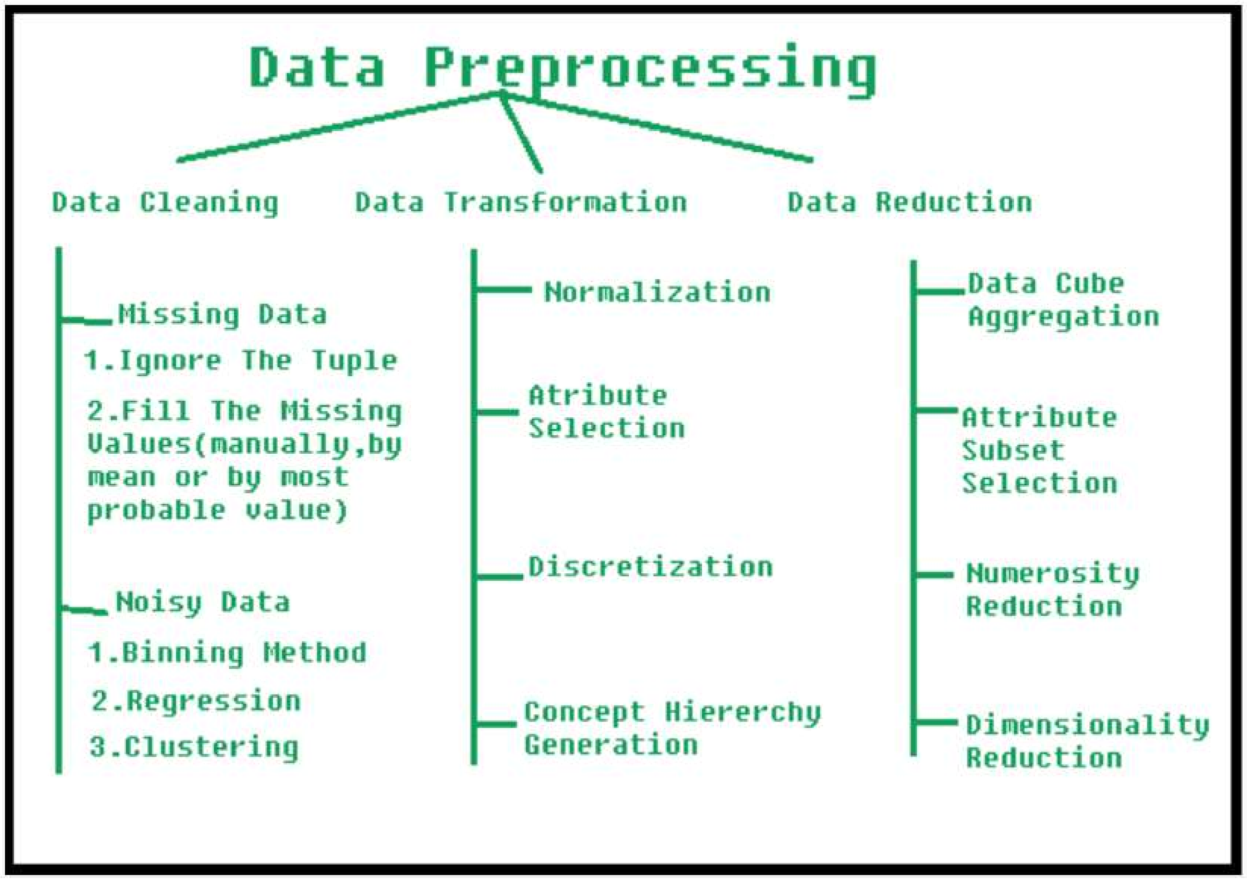
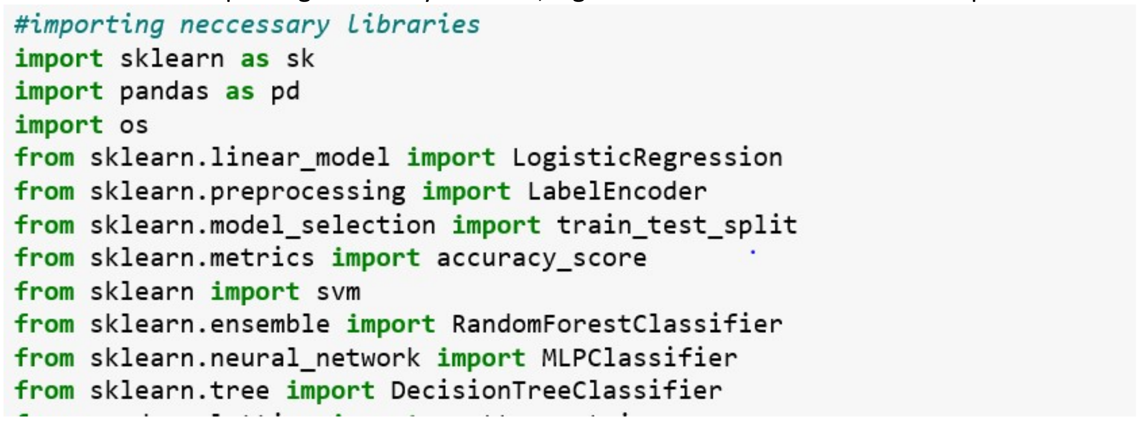


Fig5. Steps for data pre-processing (GeeksforGeeks, 2019)

2.1.1 Data Pre-processing techniques/ steps used in Q2 for Regression & Classification

The process starts with importing necessary libraries, Fig6 shows some of the libraries imported:



```
#importing neccessary libraries
import sklearn as sk
import pandas as pd
import os
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
```

Fig6. Example of importing necessary libraries

- ## Data Cleaning:

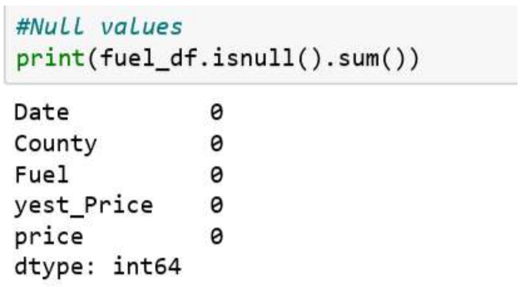
  Data cleaning is an essential step of pre-processing, which could be achieved by eliminating "Missing Data," followed by dealing with "noisy data" for better and improved results. (GeeksforGeeks, 2019)

  For both regression and classification projects, data cleaning is achieved by:

  1. Checking for missing values
  2. Eliminating missing values if found

  ### Regression:

  For the regression dataset, there were no missing values.



```
#Null values
print(fuel_df.isnull().sum())
```

```
Date          0
County        0
Fuel          0
yest_Price    0
price         0
dtype: int64
```

Fig7. Number of missing values for regression project

Classification:

For the classification project, there were few missing values present, and were eliminated, as seen in Fig9.

```
#Data Cleaning
sns.heatmap(test1.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2726ca2c948>
```

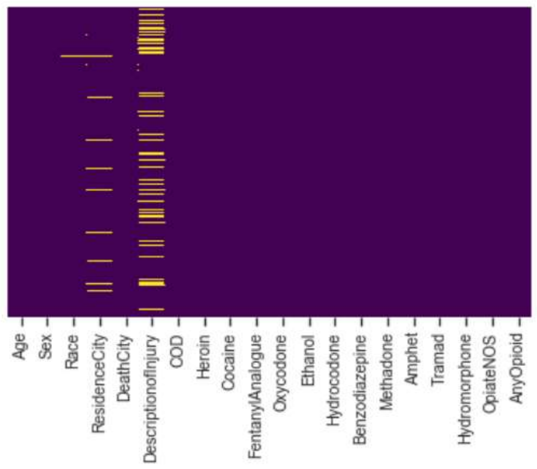
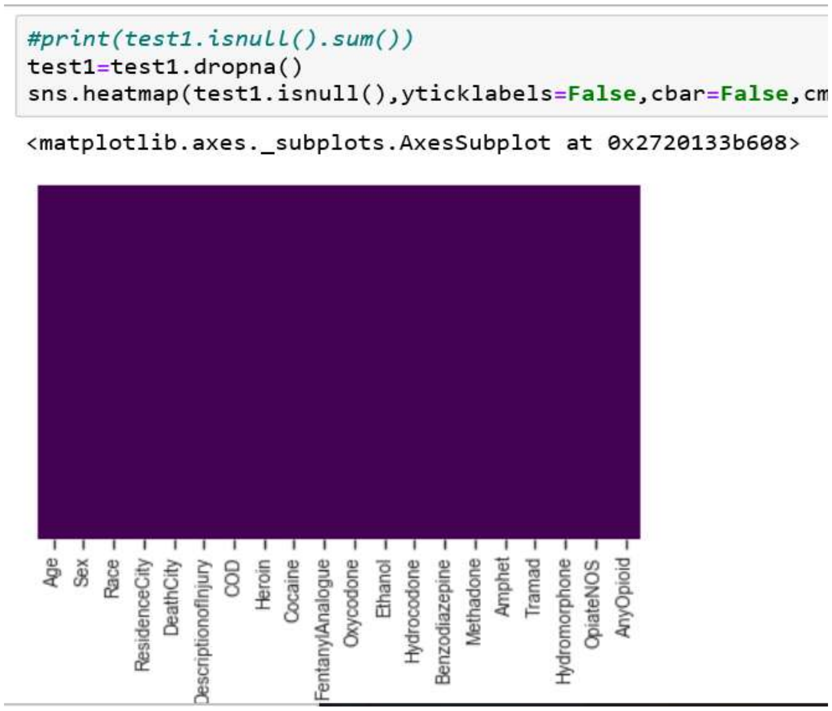


Fig8. Heatmap of missing values for classification project

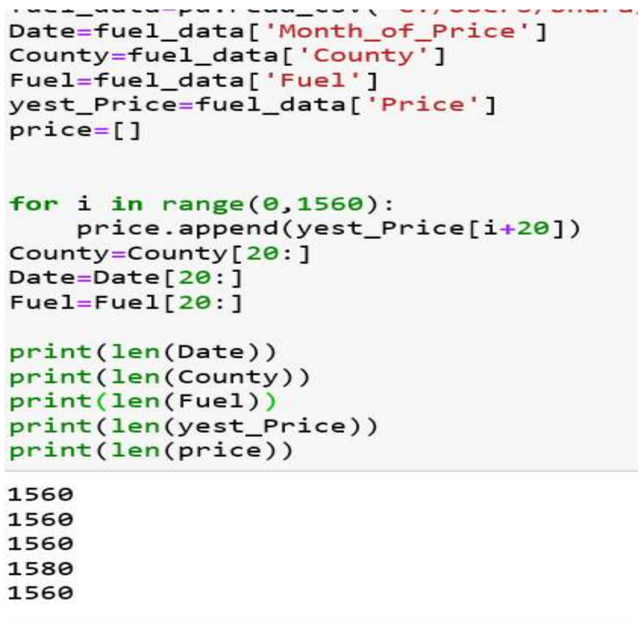


```
#print(test1.isnull().sum())
test1=test1.dropna()
sns.heatmap(test1.isnull(),yticklabels=False,cbar=False,cm
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2720133b608>
```

Fig9. Missing Values dropped (Classification Project)

- ## 2. Data Transformation and Attribute Subset Selection:
  Commonly used data transformation methods are Normalization, Data Reduction, Discretization, and Concept Hierarchy Generation. (GeeksforGeeks, 2019)
  ### Regression:
  A new attribute called yest_price was created from the existing data, as shown in Fig10.

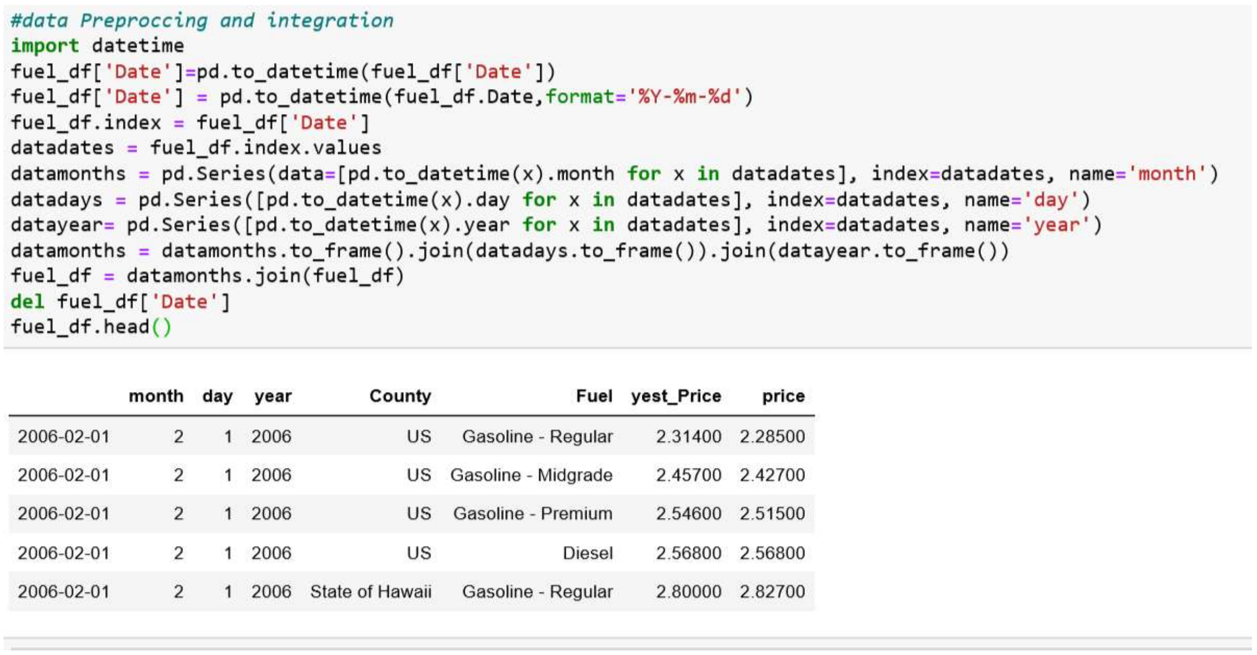


```
Date=fuel_data['Month_of_Price']
County=fuel_data['County']
Fuel=fuel_data['Fuel']
yest_Price=fuel_data['Price']
price=[]


for i in range(0,1560):
     price.append(yest_Price[i+20])
County=County[20:]
Date=Date[20:]
Fuel=Fuel[20:]

print(len(Date))
print(len(County))
print(len(Fuel))
print(len(yest_Price))
print(len(price))
```

```
1560
1560
1560
1580
1560
```

Fig10. Creating a new attribute

The column "date" was split into day, month, and year which is a common practice for time series prediction.

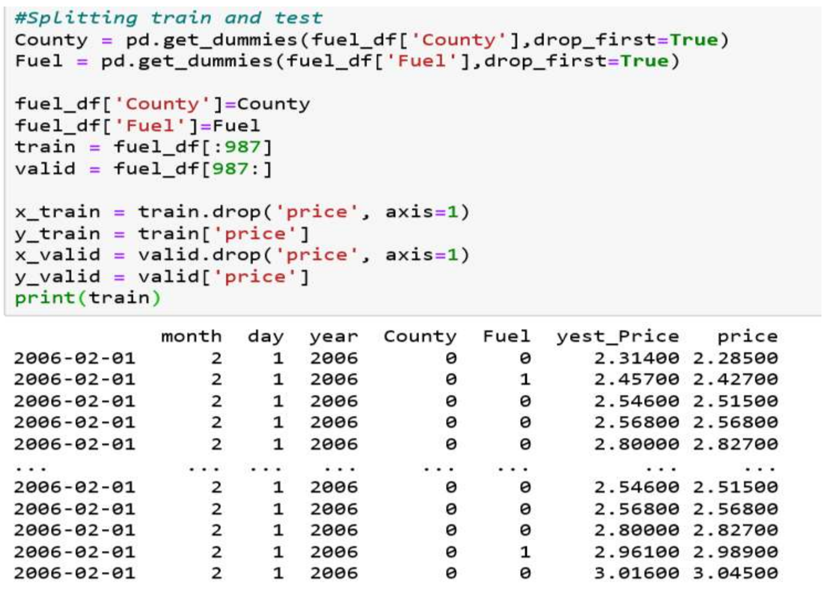


```python
#data Preproccing and integration
import datetime
fuel_df['Date']=pd.to_datetime(fuel_df['Date'])
fuel_df['Date'] = pd.to_datetime(fuel_df.Date,format='%Y-%m-%d')
fuel_df.index = fuel_df['Date']
datadates = fuel_df.index.values
datamonths = pd.Series(data=[pd.to_datetime(x).month for x in datadates], index=datadates, name='month')
datadays = pd.Series([pd.to_datetime(x).day for x in datadates], index=datadates, name='day')
datayear= pd.Series([pd.to_datetime(x).year for x in datadates], index=datadates, name='year')
datamonths = datamonths.to_frame().join(datadays.to_frame()).join(datayear.to_frame())
fuel_df = datamonths.join(fuel_df)
del fuel_df['Date']
fuel_df.head()
```

	month	day	year	County	Fuel	yest_Price	price
2006-02-01	2	1	2006	US	Gasoline - Regular	2.31400	2.28500
2006-02-01	2	1	2006	US	Gasoline - Midgrade	2.45700	2.42700
2006-02-01	2	1	2006	US	Gasoline - Premium	2.54600	2.51500
2006-02-01	2	1	2006	US	Diesel	2.56800	2.56800
2006-02-01	2	1	2006	State of Hawaii	Gasoline - Regular	2.80000	2.82700

Fig11. Splitting column date

Followed by normalizing nominal attributes, as shown in Fig8.



```python
#Splitting train and test
County = pd.get_dummies(fuel_df['County'],drop_first=True)
Fuel = pd.get_dummies(fuel_df['Fuel'],drop_first=True)

fuel_df['County']=County
fuel_df['Fuel']=Fuel
train = fuel_df[:987]
valid = fuel_df[987:]

x_train = train.drop('price', axis=1)
y_train = train['price']
x_valid = valid.drop('price', axis=1)
y_valid = valid['price']
print(train)
```

```
            month  day  year  County  Fuel  yest_Price   price
2006-02-01      2    1  2006       0     0     2.31400 2.28500
2006-02-01      2    1  2006       0     1     2.45700 2.42700
2006-02-01      2    1  2006       0     0     2.54600 2.51500
2006-02-01      2    1  2006       0     0     2.56800 2.56800
2006-02-01      2    1  2006       0     0     2.80000 2.82700
...           ...  ...   ...     ...   ...         ...     ...
2006-02-01      2    1  2006       0     0     2.54600 2.51500
2006-02-01      2    1  2006       0     0     2.56800 2.56800
2006-02-01      2    1  2006       0     0     2.80000 2.82700
2006-02-01      2    1  2006       0     1     2.96100 2.98900
2006-02-01      2    1  2006       0     0     3.01600 3.04500
```

Fig12 Normalizing data(Regression Classification Project)

There was no feature selection technique used for this project as all the attributes were highly co-related as seen in Fig13

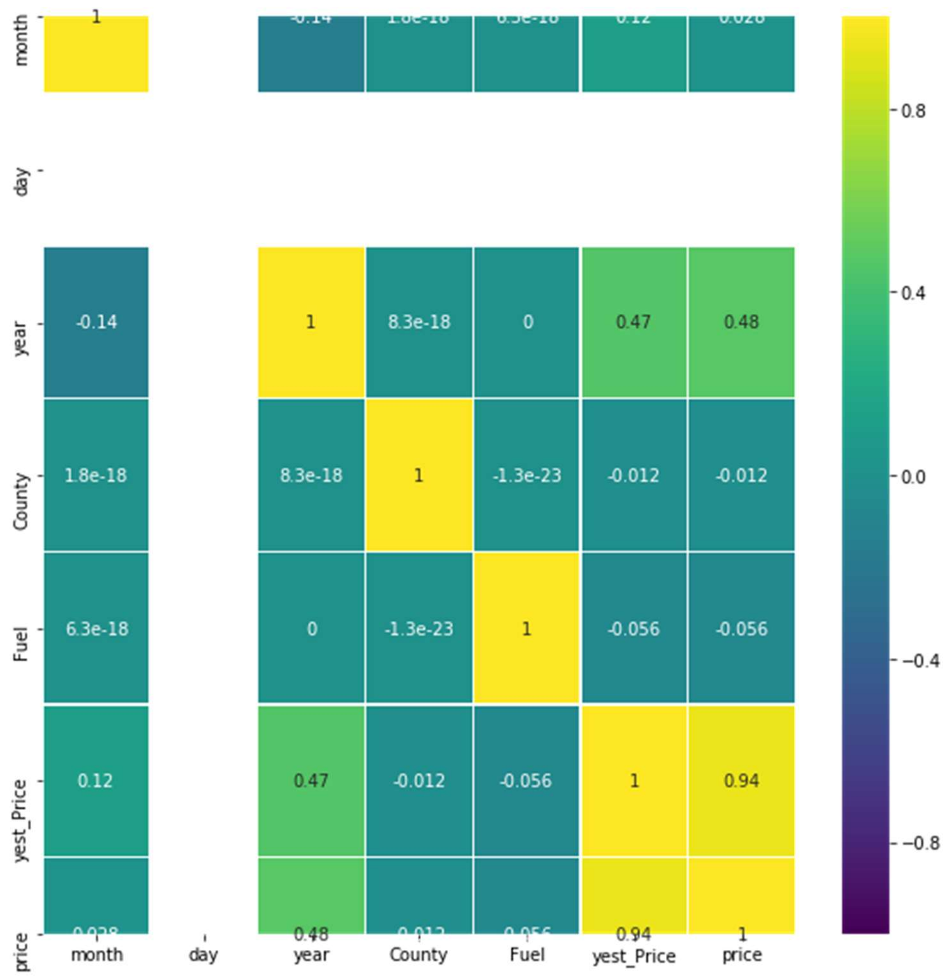
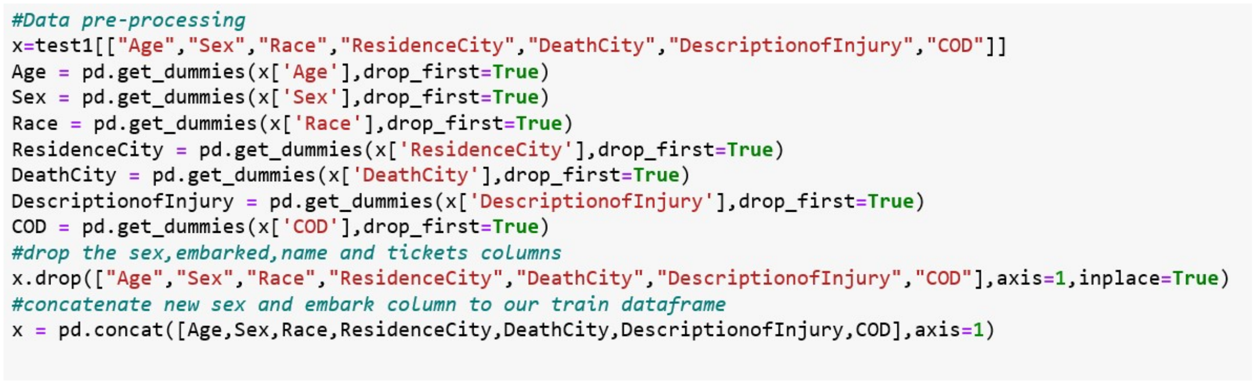


Fig13. Correlation Matrix

Classification:

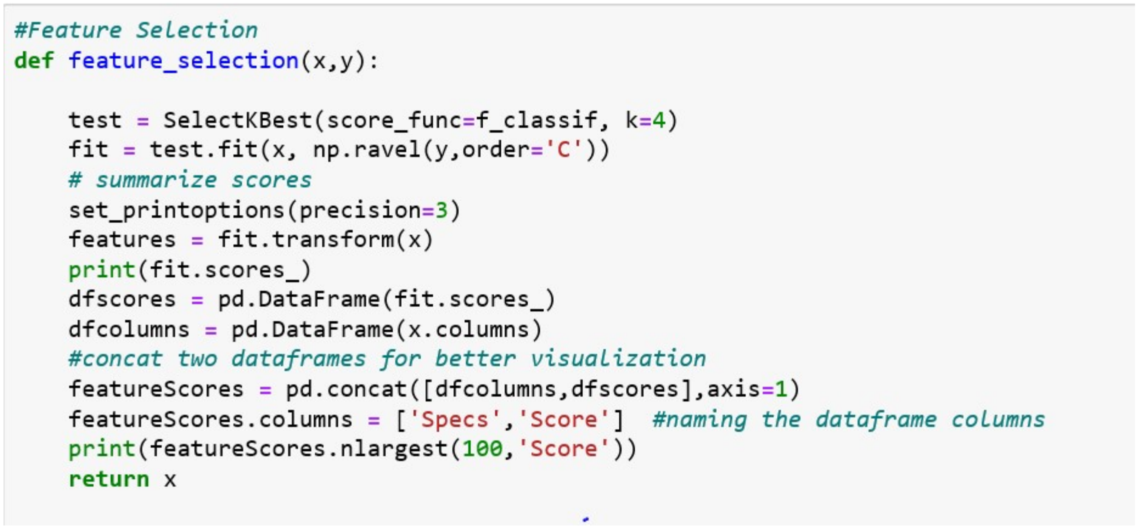
Discretization is done for attributes containing nominal data using pandas function dummy.

Followed by feature selection using Seleckbest.



```
#Data pre-processing
x=test1[["Age","Sex","Race","ResidenceCity","DeathCity","DescriptionofInjury","COD"]]
Age = pd.get_dummies(x['Age'],drop_first=True)
Sex = pd.get_dummies(x['Sex'],drop_first=True)
Race = pd.get_dummies(x['Race'],drop_first=True)
ResidenceCity = pd.get_dummies(x['ResidenceCity'],drop_first=True)
DeathCity = pd.get_dummies(x['DeathCity'],drop_first=True)
DescriptionofInjury = pd.get_dummies(x['DescriptionofInjury'],drop_first=True)
COD = pd.get_dummies(x['COD'],drop_first=True)
#drop the sex,embarked,name and tickets columns
x.drop(["Age","Sex","Race","ResidenceCity","DeathCity","DescriptionofInjury","COD"],axis=1,inplace=True)
#concatenate new sex and embark column to our train dataframe
x = pd.concat([Age,Sex,Race,ResidenceCity,DeathCity,DescriptionofInjury,COD],axis=1)
```

Fig14. Discretization attributes(Classification Project)



```
#Feature Selection
def feature_selection(x,y):

    test = SelectKBest(score_func=f_classif, k=4)
    fit = test.fit(x, np.ravel(y,order='C'))
    # summarize scores
    set_printoptions(precision=3)
    features = fit.transform(x)
    print(fit.scores_)
    dfscores = pd.DataFrame(fit.scores_)
    dfcolumns = pd.DataFrame(x.columns)
    #concat two dataframes for better visualization
    featureScores = pd.concat([dfcolumns,dfscores],axis=1)
    featureScores.columns = ['Specs','Score']  #naming the dataframe columns
    print(featureScores.nlargest(100,'Score'))
    return x
```

Fig15. Feature Selection for Classification

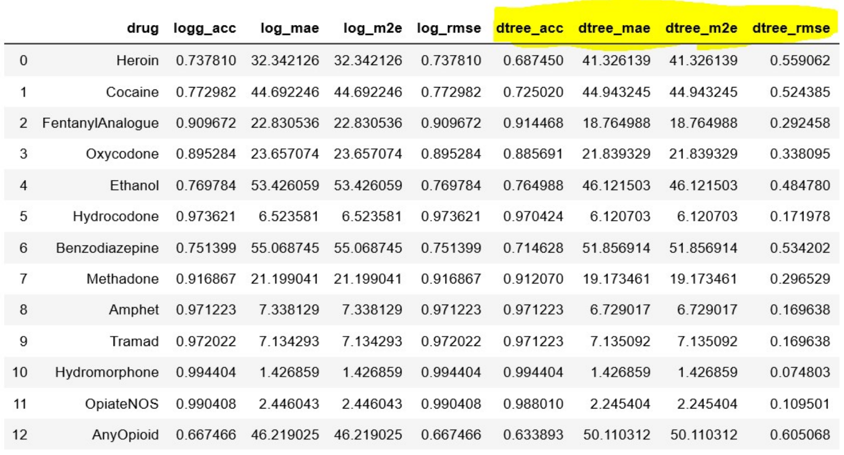
Chp. 3.

3.1 Decision Tree

Decision Trees (DTs) is a non-parametric supervised classification and regression learning technique. The aim is to build a model that predicts the value of a target variable by understanding basic decision rules based on the data characteristics. (Scikit-learn.org, 2019)

Classification:

Like other classification methods, DecisionTreeClassifier uses two arrays the first one being input: a sparse or dense [n sample, n features] array, containing training samples, and an array of Y integer values, size [n samples], containing training sample class labels. (Scikit-learn.org, 2019)



	drug	logg_acc	log_mae	log_m2e	log_rmse	dtree_acc	dtree_mae	dtree_m2e	dtree_rmse
0	Heroin	0.737810	32.342126	32.342126	0.737810	0.687450	41.326139	41.326139	0.559062
1	Cocaine	0.772982	44.692246	44.692246	0.772982	0.725020	44.943245	44.943245	0.524385
2	FentanylAnalogue	0.909672	22.830536	22.830536	0.909672	0.914468	18.764988	18.764988	0.292458
3	Oxycodone	0.895284	23.657074	23.657074	0.895284	0.885691	21.839329	21.839329	0.338095
4	Ethanol	0.769784	53.426059	53.426059	0.769784	0.764988	46.121503	46.121503	0.484780
5	Hydrocodone	0.973621	6.523581	6.523581	0.973621	0.970424	6.120703	6.120703	0.171978
6	Benzodiazepine	0.751399	55.068745	55.068745	0.751399	0.714628	51.856914	51.856914	0.534202
7	Methadone	0.916867	21.199041	21.199041	0.916867	0.912070	19.173461	19.173461	0.296529
8	Amphet	0.971223	7.338129	7.338129	0.971223	0.971223	6.729017	6.729017	0.169638
9	Tramad	0.972022	7.134293	7.134293	0.972022	0.971223	7.135092	7.135092	0.169638
10	Hydromorphone	0.994404	1.426859	1.426859	0.994404	0.994404	1.426859	1.426859	0.074803
11	OpiateNOS	0.990408	2.446043	2.446043	0.990408	0.988010	2.245404	2.245404	0.109501
12	AnyOpioid	0.667466	46.219025	46.219025	0.667466	0.633893	50.110312	50.110312	0.605068

Fig16. Decision tree results for classification project for various drugs

In Fig16. Shows accuracy(dtree_acc),Mean_Absolute_Error(dtree_mae), Mean_Squared_Error (dtree_m2e) and RMSE(dtree_rmse).

Regression:

As with Classification, in this case, the fit form is only meant to have float point values rather than integer values: Arrays X and Y as arguments. (Scikit-learn.org, 2019).

3.2 What is Entropy

Entropy, in simple words, is a "measure of disorder," Entropy helps settle on the tree boundaries by breaking the data and deciding its outer limits. (T, 2019)

Mathematical formula for Entropy: **$E(S) = -\Sigma P_k \log_2 P_k$**

3.3 What is Information Gain

It estimates Entropy or surprise reduction by somehow transforming a dataset. It is in the decision tree for building decisions from a training dataset by analyzing the data gain for each variable and by selecting a variable that maximizes information gain, thus minimizing Entropy and splitting the data set best into a group for efficient Classification. (https://www.facebook.com/MachineLearningMastery, 2019)

Information Gain = Entropy (parent) – [weight average] * Entropy (children)

Fig17. Information Gain Formula

2.4 Chinese Restaurant Process

The Chinese restaurant method (CRP), reduced to the simplest possible definition, provides a partition distribution. Suppose we have a series of comments and want to group/divide them into groups. We can use the CRP to do that, t he CRP takes its name from an end-to-end metaphor focused on chinese restaurants in San Francisco. -- potential community in this metaphor corresponds to a table in a huge Chinese restaurant. Every observation is a 'customer' who enters the restaurant and sits at the table. Through this example, people are supposed to choose to sit at common tables, but a non-zero chance of a new client sitting at a currently unoccupied table is still there. To see the way it works, assume N customers are currently in the restaurant, and allow Zi to be an indicator variable which shows us the table at which the customer sits. Therefore, we have a "table assignment" matrix, z = (z1, z2, ., zN). For example, if N = 6 was given in table at z = (1, 2, 1, 3, 4), then there were at table 1 three customers (i.e., customer 1, 2 and 4), and at table 2 (i.e. customer 3), table 3 (i.e. customer 5) and table 4 (i.e. customer 6). The actual table numbers mean nothing: the indexable variables are just convenient. That is to say, the inverse of the two are z = (1, 1, 2, 1, 3, 4) and z = (2, 2, 1, 2, 4, 3). Next, let nk indicate how many people are seated at the table of kth, and let K indicate how many non-empty tables are not. So the n= (n1,. ., nK) "counts" vector tells us how many persons each table has. If the assignment vector of the table is z = (1, 1, 1, 2, 3, 4), then the count vector is n = (3, 1, 1, 1, 1). Keep in mind PK k=1 nk = N. It's very straightforward to explain CRP now that we've adopted this terminology. When N customers are actually in the restaurant, So it is proportional to the importance of that table that customer N + 1 sits at the kth table:

P(zN+1 = k | n, α) = nk N + α

Where the α "dispersion" parameter of the CRP is called the "concentration parameter".

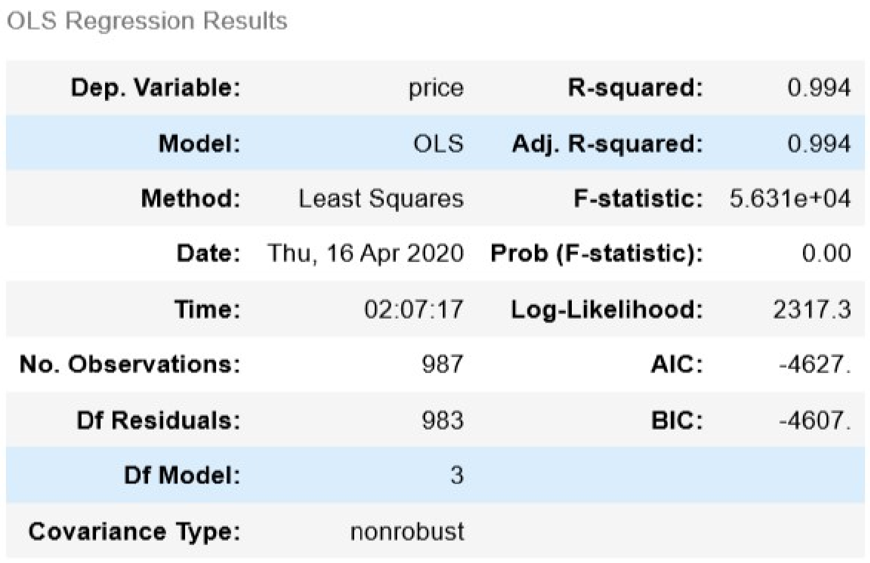
If we want to mark it as a table, P(zN+1 = K + 1 | n, α) = α N + α .

In combination with Equations 1 and 2, the Chinese restaurant cycle is described. This is the main concept behind the CRP. (Navarro and Perfors, n.d.).

Ch.4 .

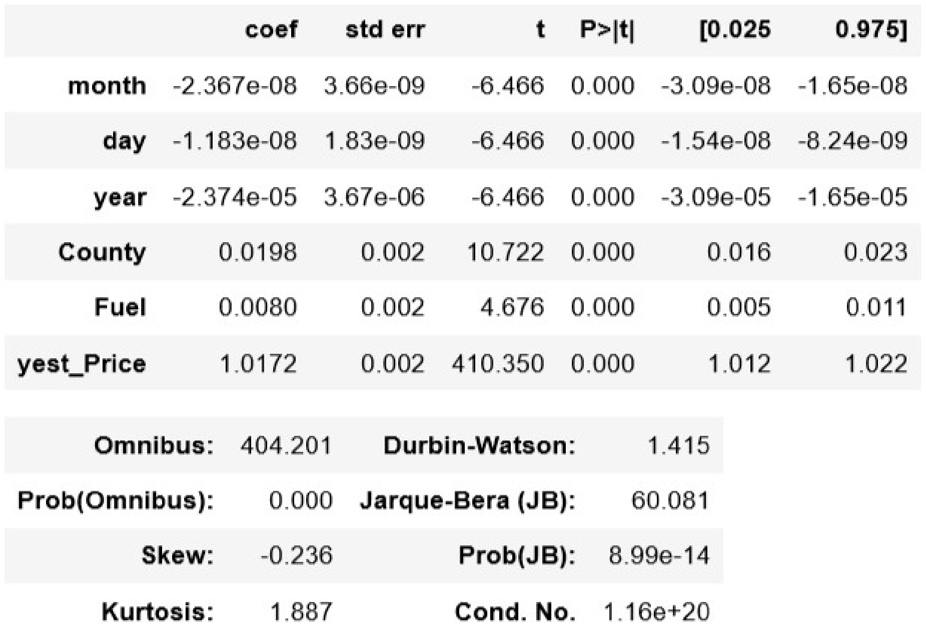
4.1 Results for Regression

For our primary objective, "forecast gasoline prices using Linear Regression," the Linear Regression model did exceptionally well with 0.994 R2 value. Fig19 displays the weights for different attributes created after Discretization.


OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.994
Model:	OLS	**Adj. R-squared:**	0.994
Method:	Least Squares	**F-statistic:**	5.631e+04
Date:	Thu, 16 Apr 2020	**Prob (F-statistic):**	0.00
Time:	02:07:17	**Log-Likelihood:**	2317.3
No. Observations:	987	**AIC:**	-4627.
Df Residuals:	983	**BIC:**	-4607.
Df Model:	3		
Covariance Type:	nonrobust		

Fig18. Results for Linear Regression on Original dataset



	coef	std err	t	P>|t|	[0.025	0.975]
month	-2.367e-08	3.66e-09	-6.466	0.000	-3.09e-08	-1.65e-08
day	-1.183e-08	1.83e-09	-6.466	0.000	-1.54e-08	-8.24e-09
year	-2.374e-05	3.67e-06	-6.466	0.000	-3.09e-05	-1.65e-05
County	0.0198	0.002	10.722	0.000	0.016	0.023
Fuel	0.0080	0.002	4.676	0.000	0.005	0.011
yest_Price	1.0172	0.002	410.350	0.000	1.012	1.022

Omnibus:	404.201	**Durbin-Watson:**	1.415
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	60.081
Skew:	-0.236	**Prob(JB):**	8.99e-14
Kurtosis:	1.887	**Cond. No.**	1.16e+20

Fig19. Weights for different attributes

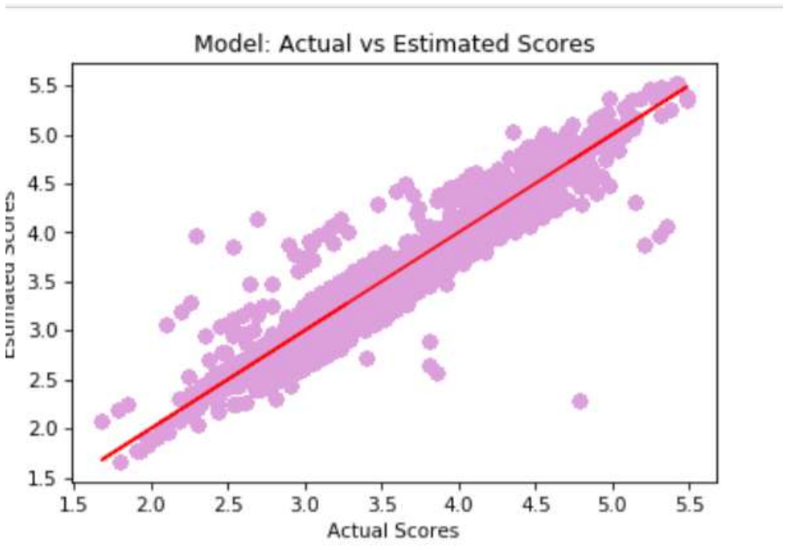
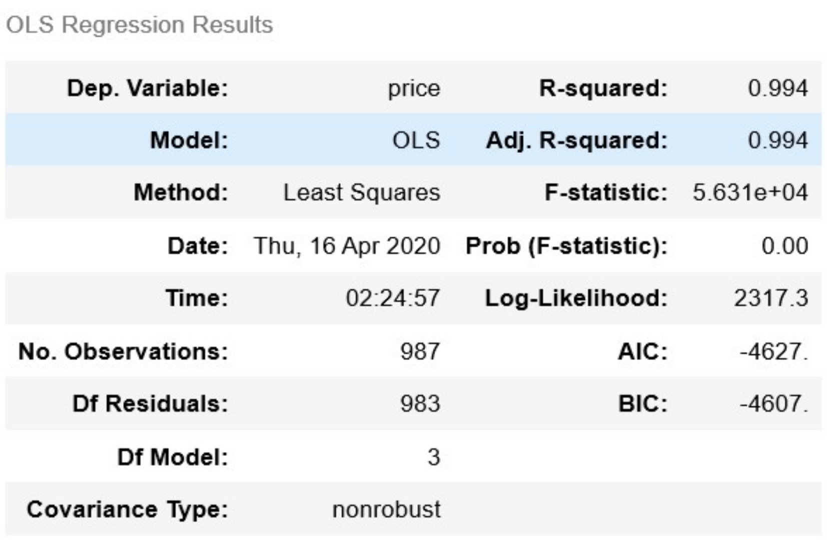


Fig20 Actual VS Estimated score comparison for Linear Regression

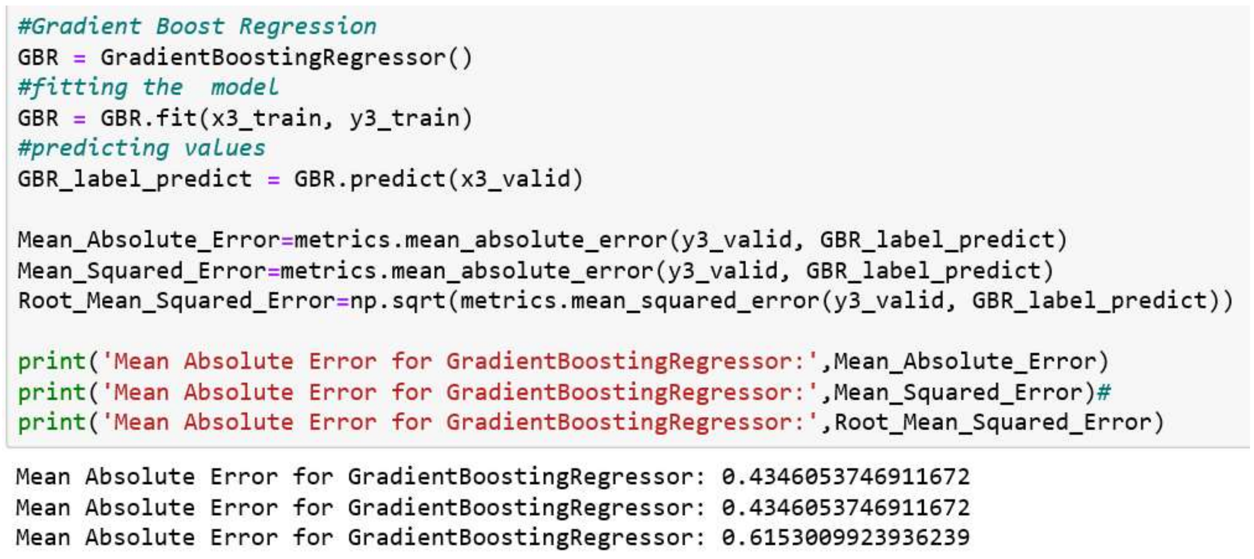
The answer to over another objective will Linear Regression on the original dataset(Consisting of Fuel type and newly created variable Yesterdays's price) would perform differently as compared with a subset of the original dataset(not containing Fuel-type and Yesterday's price) was clearly "NO." As model gave exactly the same result for both the datsets as shown in Fig21.


OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.994
Model:	OLS	**Adj. R-squared:**	0.994
Method:	Least Squares	**F-statistic:**	5.631e+04
Date:	Thu, 16 Apr 2020	**Prob (F-statistic):**	0.00
Time:	02:24:57	**Log-Likelihood:**	2317.3
No. Observations:	987	**AIC:**	-4627.
Df Residuals:	983	**BIC:**	-4607.
Df Model:	3		
Covariance Type:	nonrobust		

Fig 21. Results of Linear Regression on Subset

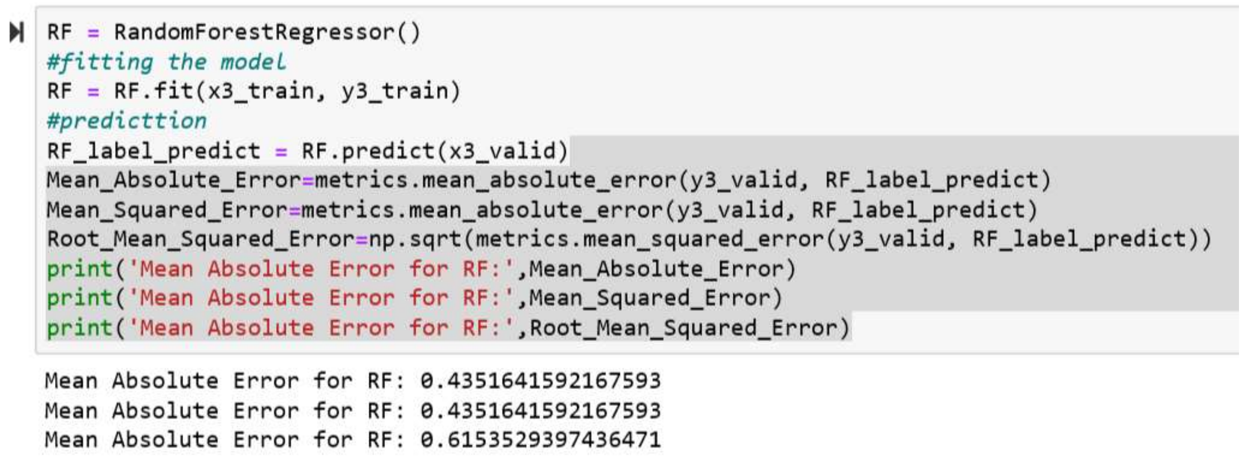
Below figures display the performance of different regressors used on Original Dataset.



```
#Gradient Boost Regression
GBR = GradientBoostingRegressor()
#fitting the  model
GBR = GBR.fit(x3_train, y3_train)
#predicting values
GBR_label_predict = GBR.predict(x3_valid)

Mean_Absolute_Error=metrics.mean_absolute_error(y3_valid, GBR_label_predict)
Mean_Squared_Error=metrics.mean_absolute_error(y3_valid, GBR_label_predict)
Root_Mean_Squared_Error=np.sqrt(metrics.mean_squared_error(y3_valid, GBR_label_predict))

print('Mean Absolute Error for GradientBoostingRegressor:',Mean_Absolute_Error)
print('Mean Absolute Error for GradientBoostingRegressor:',Mean_Squared_Error)#
print('Mean Absolute Error for GradientBoostingRegressor:',Root_Mean_Squared_Error)
```

```
Mean Absolute Error for GradientBoostingRegressor: 0.4346053746911672
Mean Absolute Error for GradientBoostingRegressor: 0.4346053746911672
Mean Absolute Error for GradientBoostingRegressor: 0.6153009923936239
```

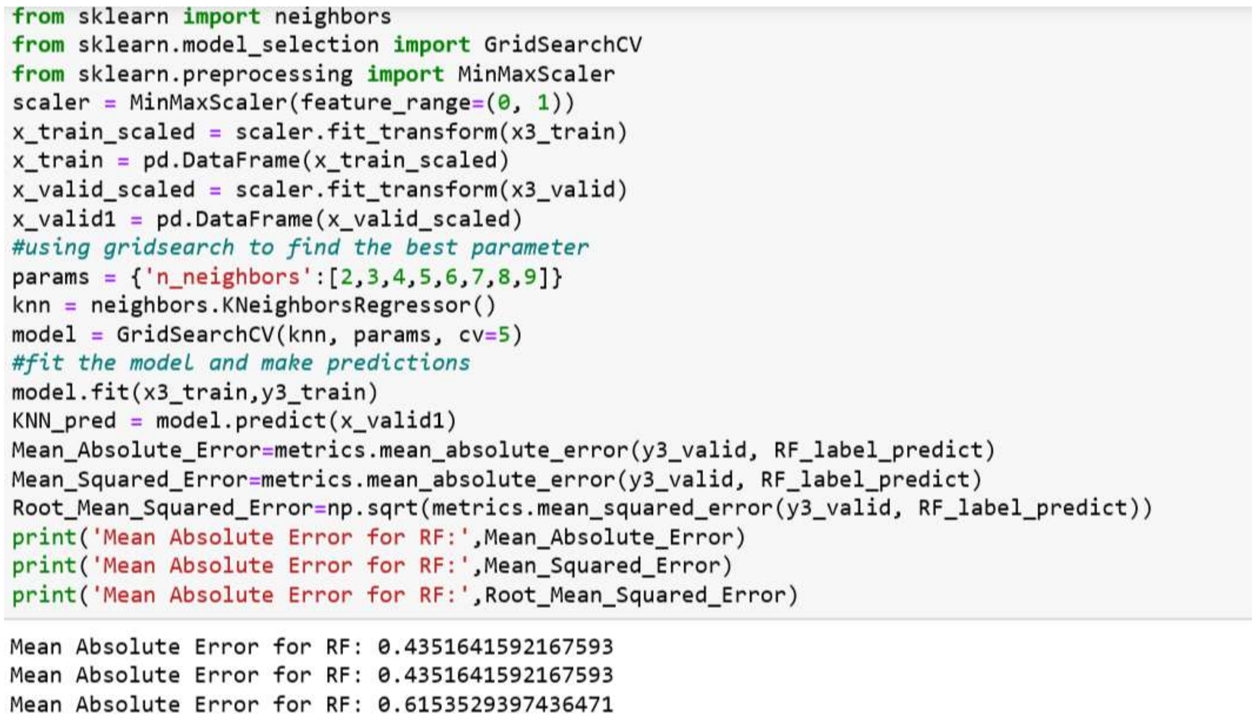
Fig 22. Results for Gradient Boost Regression



```
RF = RandomForestRegressor()
#fitting the model
RF = RF.fit(x3_train, y3_train)
#predicttion
RF_label_predict = RF.predict(x3_valid)
Mean_Absolute_Error=metrics.mean_absolute_error(y3_valid, RF_label_predict)
Mean_Squared_Error=metrics.mean_absolute_error(y3_valid, RF_label_predict)
Root_Mean_Squared_Error=np.sqrt(metrics.mean_squared_error(y3_valid, RF_label_predict))
print('Mean Absolute Error for RF:',Mean_Absolute_Error)
print('Mean Absolute Error for RF:',Mean_Squared_Error)
print('Mean Absolute Error for RF:',Root_Mean_Squared_Error)
```

```
Mean Absolute Error for RF: 0.4351641592167593
Mean Absolute Error for RF: 0.4351641592167593
Mean Absolute Error for RF: 0.6153529397436471
```

Fig 23. Results for Random Forest



```
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
x_train_scaled = scaler.fit_transform(x3_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x3_valid)
x_valid1 = pd.DataFrame(x_valid_scaled)
#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)
#fit the model and make predictions
model.fit(x3_train,y3_train)
KNN_pred = model.predict(x_valid1)
Mean_Absolute_Error=metrics.mean_absolute_error(y3_valid, RF_label_predict)
Mean_Squared_Error=metrics.mean_absolute_error(y3_valid, RF_label_predict)
Root_Mean_Squared_Error=np.sqrt(metrics.mean_squared_error(y3_valid, RF_label_predict))
print('Mean Absolute Error for RF:',Mean_Absolute_Error)
print('Mean Absolute Error for RF:',Mean_Squared_Error)
print('Mean Absolute Error for RF:',Root_Mean_Squared_Error)
```

```
Mean Absolute Error for RF: 0.4351641592167593
Mean Absolute Error for RF: 0.4351641592167593
Mean Absolute Error for RF: 0.6153529397436471
```

Fig 24.

According to the performance of all the regressors, the Linear Regression Model outshines other models by significant Margin.

4.2 Results for Correlation

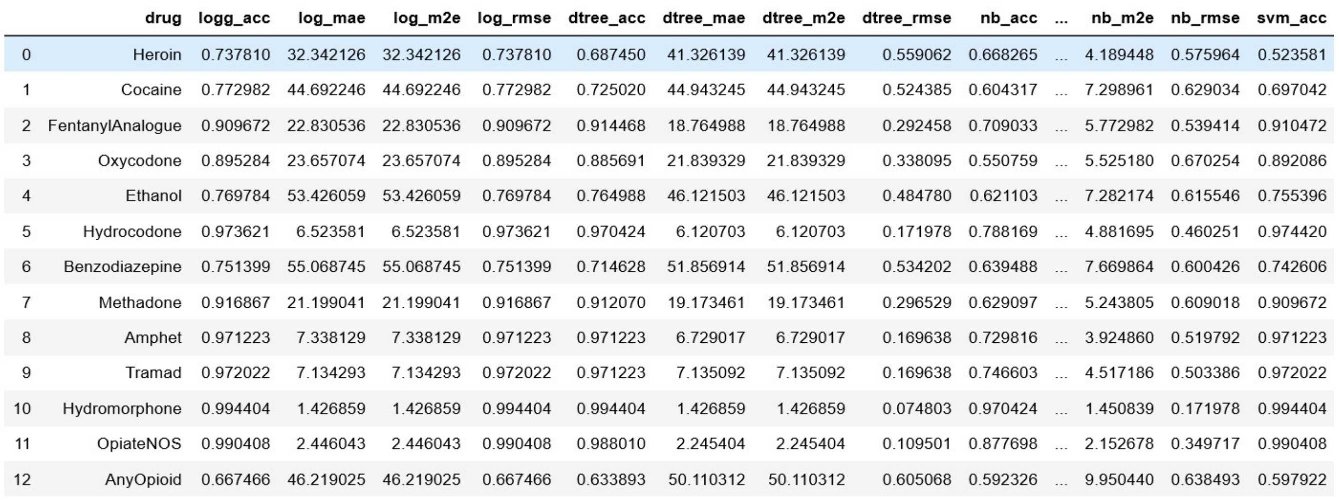
"Comparative analysis of Machine Learning Classifiers on Accidental related deaths dataset." It was a very challenging and exciting study, and most of the models performed pretty well, surprisingly, on all the drugs. Table 1 gives the Summary of Performance for all the Models on different drugs.

Abbreviations in table1:

1. logg – Logistic Regression
2. dtree- Decision Tree
3. nb- Naïve bayes
4. svm- Support Vector Machines
5. knn- k-nearest neighbors
6. acc- Accuracy
7. mae- Mean absolute error
8. m2e- Mean squared error
9. rmse- Root mean squared error

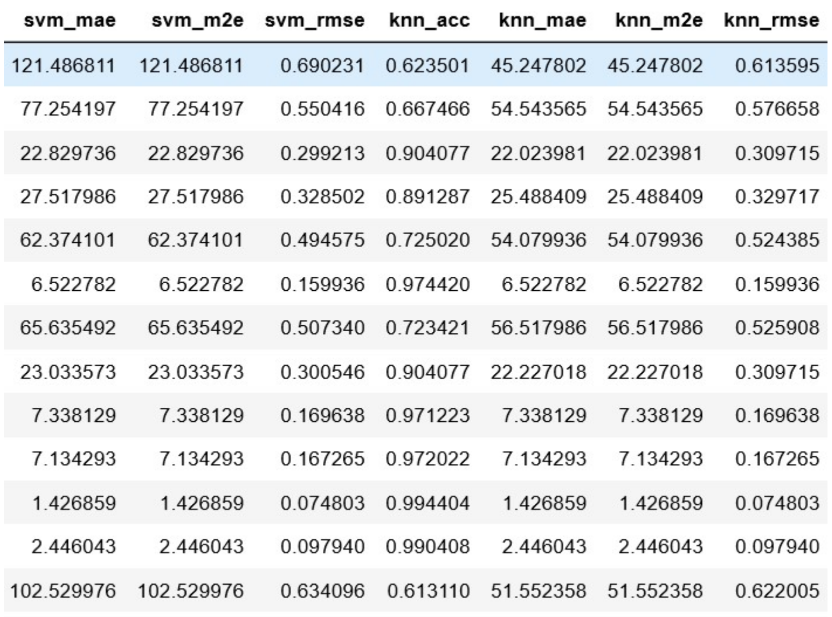
According to the result in Table 1 logistic regression was a complete package and a star performer for most of the drugs.

In the Classification project, feature selection technique and using necessary variables increased the model performance and efficiency drastically.



	drug	logg_acc	log_mae	log_m2e	log_rmse	dtree_acc	dtree_mae	dtree_m2e	dtree_rmse	nb_acc	...	nb_m2e	nb_rmse	svm_acc
0	Heroin	0.737810	32.342126	32.342126	0.737810	0.687450	41.326139	41.326139	0.559062	0.668265	...	4.189448	0.575964	0.523581
1	Cocaine	0.772982	44.692246	44.692246	0.772982	0.725020	44.943245	44.943245	0.524385	0.604317	...	7.298961	0.629034	0.697042
2	FentanylAnalogue	0.909672	22.830536	22.830536	0.909672	0.914468	18.764988	18.764988	0.292458	0.709033	...	5.772982	0.539414	0.910472
3	Oxycodone	0.895284	23.657074	23.657074	0.895284	0.885691	21.839329	21.839329	0.338095	0.550759	...	5.525180	0.670254	0.892086
4	Ethanol	0.769784	53.426059	53.426059	0.769784	0.764988	46.121503	46.121503	0.484780	0.621103	...	7.282174	0.615546	0.755396
5	Hydrocodone	0.973621	6.523581	6.523581	0.973621	0.970424	6.120703	6.120703	0.171978	0.788169	...	4.881695	0.460251	0.974420
6	Benzodiazepine	0.751399	55.068745	55.068745	0.751399	0.714628	51.856914	51.856914	0.534202	0.639488	...	7.669864	0.600426	0.742606
7	Methadone	0.916867	21.199041	21.199041	0.916867	0.912070	19.173461	19.173461	0.296529	0.629097	...	5.243805	0.609018	0.909672
8	Amphet	0.971223	7.338129	7.338129	0.971223	0.971223	6.729017	6.729017	0.169638	0.729816	...	3.924860	0.519792	0.971223
9	Tramad	0.972022	7.134293	7.134293	0.972022	0.971223	7.135092	7.135092	0.169638	0.746603	...	4.517186	0.503386	0.972022
10	Hydromorphone	0.994404	1.426859	1.426859	0.994404	0.994404	1.426859	1.426859	0.074803	0.970424	...	1.450839	0.171978	0.994404
11	OpiateNOS	0.990408	2.446043	2.446043	0.990408	0.988010	2.245404	2.245404	0.109501	0.877698	...	2.152678	0.349717	0.990408
12	AnyOpioid	0.667466	46.219025	46.219025	0.667466	0.633893	50.110312	50.110312	0.605068	0.592326	...	9.950440	0.638493	0.597922

Table.1 Results for Classifier on Various Drugs



svm_mae	svm_m2e	svm_rmse	knn_acc	knn_mae	knn_m2e	knn_rmse
121.486811	121.486811	0.690231	0.623501	45.247802	45.247802	0.613595
77.254197	77.254197	0.550416	0.667466	54.543565	54.543565	0.576658
22.829736	22.829736	0.299213	0.904077	22.023981	22.023981	0.309715
27.517986	27.517986	0.328502	0.891287	25.488409	25.488409	0.329717
62.374101	62.374101	0.494575	0.725020	54.079936	54.079936	0.524385
6.522782	6.522782	0.159936	0.974420	6.522782	6.522782	0.159936
65.635492	65.635492	0.507340	0.723421	56.517986	56.517986	0.525908
23.033573	23.033573	0.300546	0.904077	22.227018	22.227018	0.309715
7.338129	7.338129	0.169638	0.971223	7.338129	7.338129	0.169638
7.134293	7.134293	0.167265	0.972022	7.134293	7.134293	0.167265
1.426859	1.426859	0.074803	0.994404	1.426859	1.426859	0.074803
2.446043	2.446043	0.097940	0.990408	2.446043	2.446043	0.097940
102.529976	102.529976	0.634096	0.613110	51.552358	51.552358	0.622005

Table.1(Continued)

Chp.5.

Conclusion

Both Regression project and Classification project gave outstanding results and answered all the stated goals/objectives. I have selected different datasets for both the project as I was always interested in learning time forecasting, so choose a time series dataset for regression. On the other hand, the "accidental related drugs" dataset was an exciting yet challenging dataset to work on, which motivated me to try and set a challenging goal for Classification. For future study, I would want to try LSTM on regression dataset and for Classification project, I would create an API or a Web-App .

References

Data.gov. (2012). *Accidental Drug Related Deaths 2012-2018 - Data.gov*. [online] Available at: https://catalog.data.gov/dataset/accidental-drug-related-deaths-january-2012-sept-2015 [Accessed 16 Apr. 2020].

Press, G. (2016). Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. *Forbes*. [online] 23 Mar. Available at: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#ae49d0b6f637 [Accessed 15 Apr. 2020].

Techopedia.com. (2020). *What is Data Preprocessing? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/14650/data-preprocessing [Accessed 15 Apr. 2020].

GeeksforGeeks. (2019). *Data Preprocessing in Data Mining - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/ [Accessed 15 Apr. 2020].

Scikit-learn.org. (2019). *1.10. Decision Trees — scikit-learn 0.22.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/tree.html [Accessed 15 Apr. 2020].

T, S. (2019). *Entropy: How Decision Trees Make Decisions - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8 [Accessed 15 Apr. 2020].

https://www.facebook.com/MachineLearningMastery (2019). *Information Gain and Mutual Information for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/information-gain-and-mutual-information/ [Accessed 16 Apr. 2020].

Navarro, D. and Perfors, A. (n.d.). *The Chinese restaurant process COMPSCI 3016: Computational Cognitive Science*. [online] Available at: https://djnavarro.github.io/compcogsci-3016/technote_chineserestaurantprocesses.pdf.