32513 - Advanced Data Analytics Algorithms

Assignment -2 : Data Analytics Project

Subject Coordinator: Dr. Jun Li

Submitted by :

Student Name: Vishnu Mohan Eda
la Dhanraj

Student ID Number: 13069909

Table of Contents

BUSINESS UNDERSTANDING

Every Nation in this world does concern about safety and welfare of every individual in their country. The major safety measures that has to be taken in this fast phased moving world is that safety while driving on roads. There are a lot of accidents and mishaps happening every day which is being recorded. These data collected not only serve for record making purposes but also give a lot of insights about the precautions to be taken to avoid them in the near future. Before divulging actually into the business problem, lets discuss about the actual business happening around it . The stakeholders to be considered out of it are Police Department, Medical Insurance provides, Road assistance departments, Ambulance Services, Car Manufacturers and Service providers and finally the government. This project mainly focuses on providing insights to the government to make necessary changes with respect to the existing safety measures or to make calculated investments on road constructions or to restrict certain old vehicles or to alter speed limit for the accident prone zone.

The major insights to be taken are , • The most common manoeuvre that people tend to make or involved in a lot of accidents . This will give a detailed view to the government to have clear sign boards to the drivers for safety purposes and assist vehicle manufacturers for designing safety precautionary items in their vehicles. • Even though these data are open and available for public use, not many will be working on them to convert and predict or use these information for betterment. This project mainly focus on providing predictions and insights which can help any of the government officials to create general awareness about the accidents that may reach people further in an effective way.

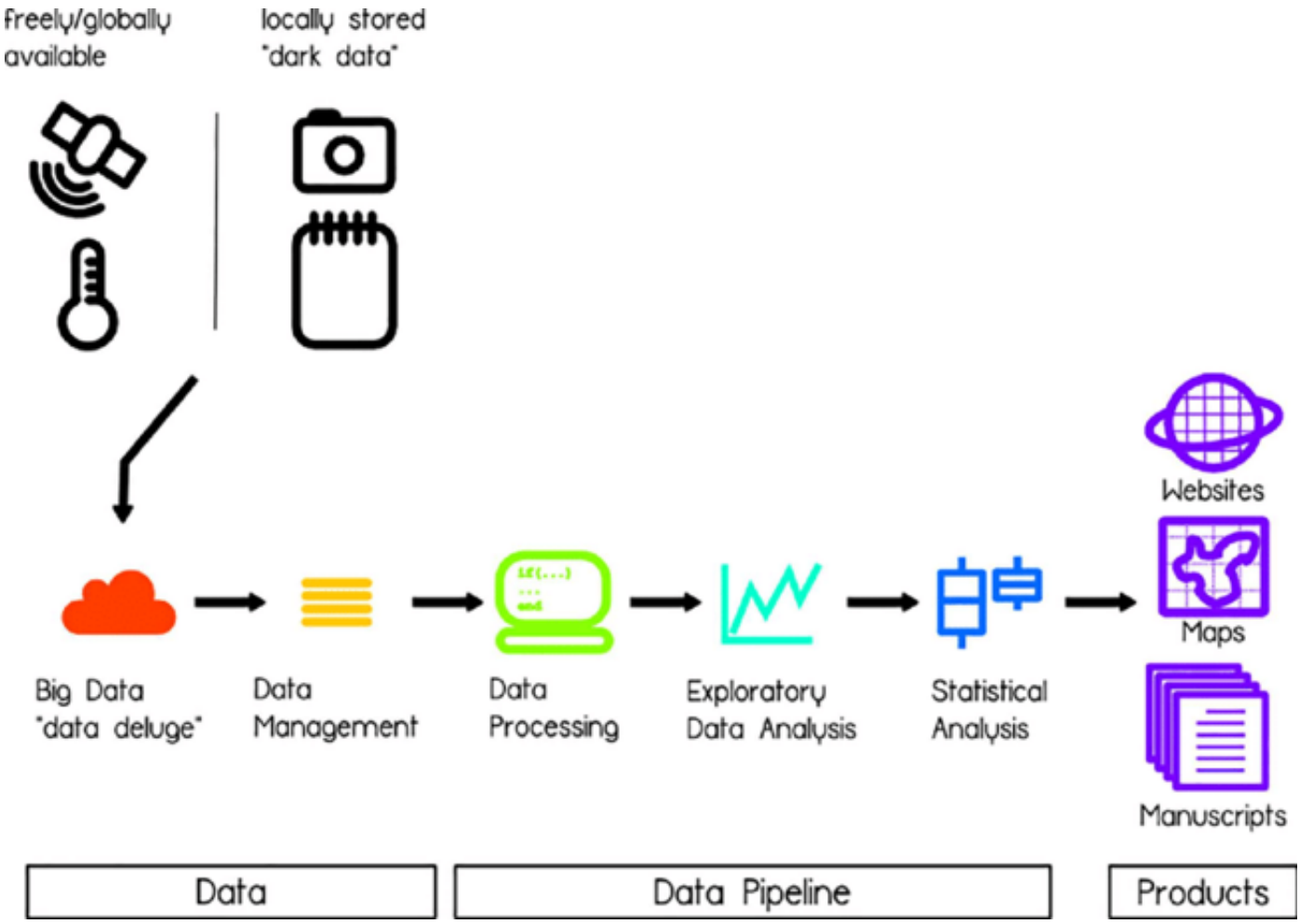
Therefore the business problem mainly focuses on “Predicting Accident_Severity to Minimise Accidents and provide safe Environment for Travel purposes” . This can be further sub divided into a series of sublevels problems that people experience as shown below,

• Whether the Road or highway is safe for Travelling purposes? • Whether the Vehicle is safe and good to travel? • The level of impact that vehicle experience ? • The level of impact of the passenger involved in the accident?

These analysis helps the Government officials or respective Stakeholders in taking measures or providing assistance as and when needed. To Perform these tasks and address all the problems stated above a dataset which has values recorded over a period of time is being considered.

DATA MINING

The Theoretical framework being employed in this project is being represented as shown in the following graph,



Once an accident occurs, the primary attributes to look out for are, • Severity of the accident i.e. Slight or serious. • No of Causalities.

The Data Chooses to work on is from the Open Data website of the UK government: OPEN DATA. These Data are mostly consists of records that has happened in the public and being recorded to the police or the government officials. Analysing them and drawing necessary insights can help them to take necessary precautions.

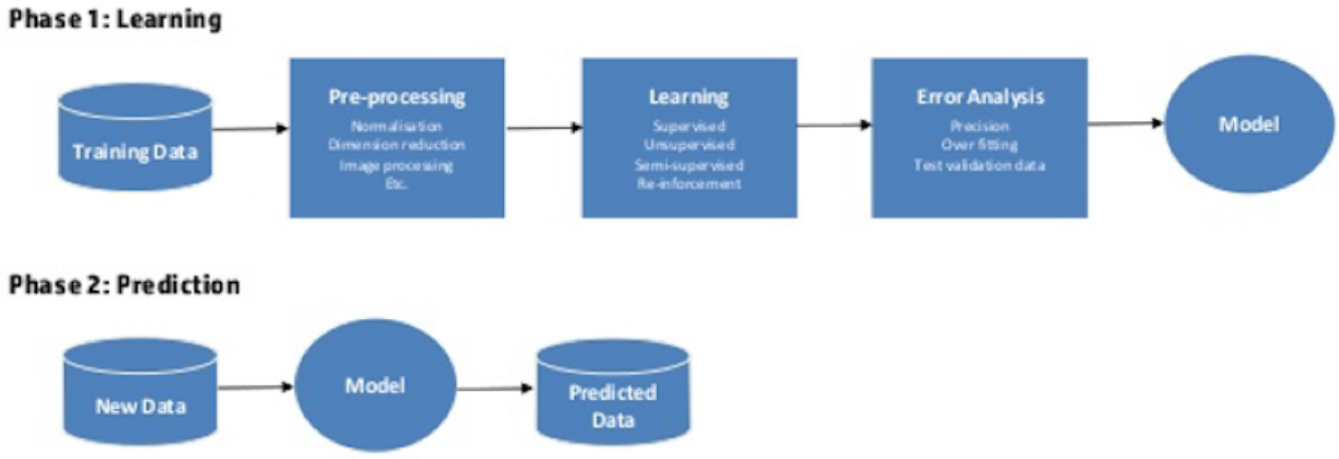
With the Data available, the main goal is to predict the severity of the accident and necessary steps to be taken in the due course. This is almost similar to the famous Dataset that we can find on the internet " TITANIC DATASET - Predicting the Survival". This data is being analysed on various parameters and finally predict whether the person has survived or not. The Data has been cleaned , categorised, visualised and predicted as stated. A detailed report of the same will be shown in the forthcoming sections.

DATA MINING PROBLEM

The Data Analysis part of the problem has been compromised into two phases as shown from the following figure.

• Data Conditioning Phase: This part mainly focuses on transforming noisy raw data and classify them as needed and make it as high quality data to be feasible for the predictor models.

• Predictive Analysis Phase: To analyse, evaluate and generate a predictive model that provides phenomenal results based on set of observations extracted from the original Raw data.



IMPLEMENTATION

The Road Safety Data extracted from the UK government Open source consists of entries recorded from 2005 - 2017. This data set comprises of two csv files.

1. Accident_Information : This File has attributes that are related to accident. Some of the important attributes in Accident_information file are ,

   - Accident_Severity - Severity of the accident _ slight or Serious/Fatal
   - Day_of_Week- Occurrence of Accident in the week
   - Road_Surface_Conditions - Condition of the Road when accident Happened . (Wet or Dry)
   - Road_Type - Type of the Way (single way or double carriage way)
   - Time- Time of occurrence of the event.
   - Weather_Conditions- Condition of Climate during the accident.
   - Urban_or_Rural - Area of the incident.

1. Vehicle_Information: This file has attributes that are related to the vehicle involved in an unique accident. Some of the important attributes in the Vehicle_informatuon file are,

   • Age_Band_of_Driver - Driver's Age.

   • Engine*Capacity*.CC.- Engine Capacity involved in the accident.

   • Vehicle_Manoeuvre - Movement of the vehicle during the incident.

   • X1st_Point_of_Impact - position of impact of the vehicle during accident.

Accident_Index:

Unique ID which carries values of an accident with respect to its location , type, Vehicle involved and other attributes. Both these data files carries attribute Accident_Index through which both accident and vehicle involved can be related.

DATA MINING PROBLEMS

• Poor Quality of Data Such as Noisy data, Missing values or not Known information found in the extracted data.

• Redundant data from different sources and forms such as text, numerical values etc..

• The data set comprises of two different csv files with accident as well vehicle information loaded. The amount of data is huge and so the efficiency and stability of data mainly depends on the selective distributive approaches.

DATA PREPARATION

Data Cleaning: Preparing the Data for computations forms the integral part of the data analytical project. After having a view on all the attributes and proper understanding of their contribution to the data set, some of the attributes of the dataset are left out on both.

The accident Data set had 34 columns as shown below,

```
In [4]: accidents_info = pd.read_csv('/Users/vishnumohan/Downloads/ML_DATASET/Accident_Information.csv'

In [5]: vehicles_info = pd.read_csv("/Users/vishnumohan/Downloads/ML_DATASET/Vehicle_Information.csv",

In [6]: accidents_info.head()
```

Out[6]:

click to scroll output; double click to hide

	Accident_Index	1st_Road_Class	1st_Road_Number	2nd_Road_Class	2nd_Road_Number	Accident_Severity	Carriageway_Haz
0	200501BS00001	A	3218.0	NaN	0.0	Serious	N
1	200501BS00002	B	450.0	C	0.0	Slight	N
2	200501BS00003	C	0.0	NaN	0.0	Slight	N
3	200501BS00004	A	3220.0	NaN	0.0	Slight	N
4	200501BS00005	Unclassified	0.0	NaN	0.0	Slight	N

5 rows × 34 columns

the vehicle Data set had 24 Columns respectively. The Accident_index forms the common attribute.

```
In [7]: vehicles_info.head()
```

Out[7]:

	Accident_Index	Age_Band_of_Driver	Age_of_Vehicle	Driver_Home_Area_Type	Driver_IMD_Decile	Engine_Capacity_.CC.	Hit_O
0	200401BS00001	26 - 35	3.0	Urban area	4.0	1588.0	
1	200401BS00002	26 - 35	NaN	Urban area	3.0	NaN	
2	200401BS00003	26 - 35	4.0	Data missing or out of range	NaN	998.0	
3	200401BS00003	66 - 75	NaN	Data missing or out of range	NaN	NaN	
4	200401BS00004	26 - 35	1.0	Urban area	4.0	124.0	

5 rows × 24 columns

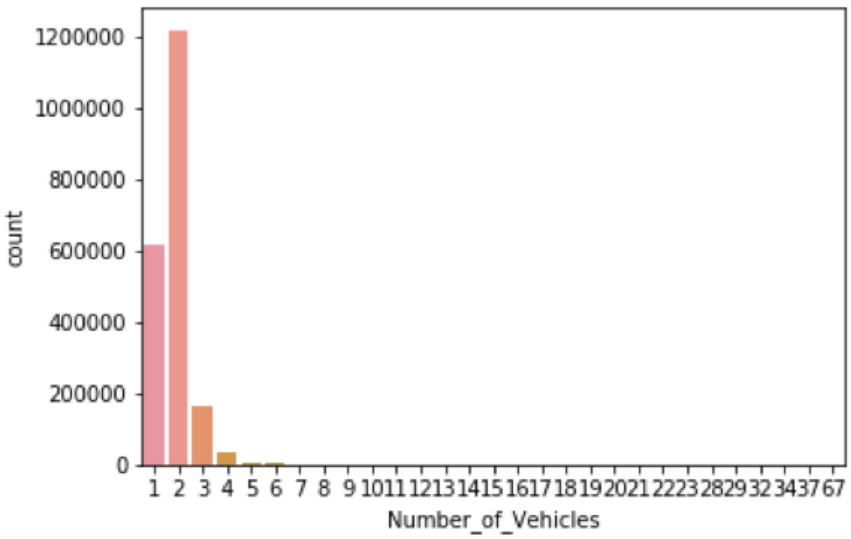
Both the Datasets are now loaded on .

The combined data set is huge and therefore the missing values, duplicate entries of certain attributes are deleted and replaced with their mean values at certain places. Also few attributes are empty or unknow those are discarded during the actual process of computations.

OUTLIERS

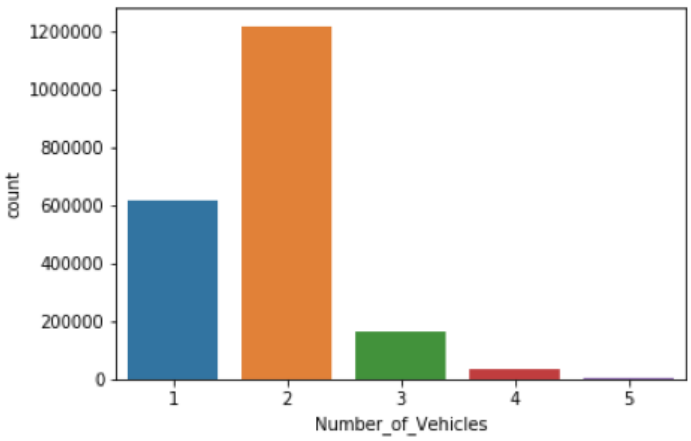
The Accident_information data set has some redundant data or outliers in it. The below Graph shows the number of vehicles involved in the accident , and there are few entries involving 11, 24, 37 vehicles involved in a single accident which is not possible.

```
In [18]: sns.countplot(accidents_info['Number_of_Vehicles'])
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1aa68710>
```



Hence considering Vehicles of countless than or equal to 5 , to be involved in the accident and removing other outliers the graph can be illustrated as shown,

```
In [19]: accidents_info = accidents_info[accidents_info.Number_of_Vehicles.astype(int) <= 5]
In [20]: sns.countplot(accidents_info['Number_of_Vehicles'])
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x118224080>
```



It is evident that most of the accidents that happened over involved only two vehicles. These information can help to regulate certain rules or speed limits in those accident prone zone.

Converting Date & Time Column

The Data and time Column is not in the right and required format. Hence the data has been loaded and converted to the required format as shown,

In [41]:
```
accidents_info['Date']= pd.to_datetime(accidents_info['Date'], format="%Y-%m-%d")
```

Firstly, the Data was sliced into first and second string from time column and then converted to a new column which is of numeric datatype. The null values in the column are dropped and then they are casted to integer values as shown,

In [10]:
```
accidents_info['Hour'] = accidents_info['Time'].str[0:2]

accidents_info['Hour'] = pd.to_numeric(accidents_info['Hour'])

accidents_info = accidents_info.dropna(subset=['Hour'])

accidents_info['Hour'] = accidents_info['Hour'].astype('int')
```

Then a Function Time_period is defined which is to categorise the hours into groups based on its time period,

In [11]:
```
def Time_period(hour):
    if hour >= 5 and hour < 10:
        return "morning Hours (5-10)"
    elif hour >= 10 and hour < 15:
        return "office hours (10-15)"
    elif hour >= 15 and hour < 19:
        return "afternoon Hours (15-19)"
    elif hour >= 19 and hour < 23:
        return "evening Hours (19-23)"
    else:
        return "night Hours (23-5)"
```

Thus the Hour has been now converted into subsequent time groups named Time_period. The data can provide the generalised column to categorise the accidents that happened over a particular period of time as shown,

In [12]:
```
accidents_info['Time_Strap'] = accidents_info['Hour'].apply(Time_period)
accidents_info[['Time', 'Hour', 'Time_Strap']].head()
```

Out[12]:

	Time	Hour	Time_Strap
0	17:42	17	afternoon Hours (15-19)
1	17:36	17	afternoon Hours (15-19)
2	00:15	0	night Hours (23-5)
3	10:35	10	office hours (10-15)
4	21:13	21	evening Hours (19-23)

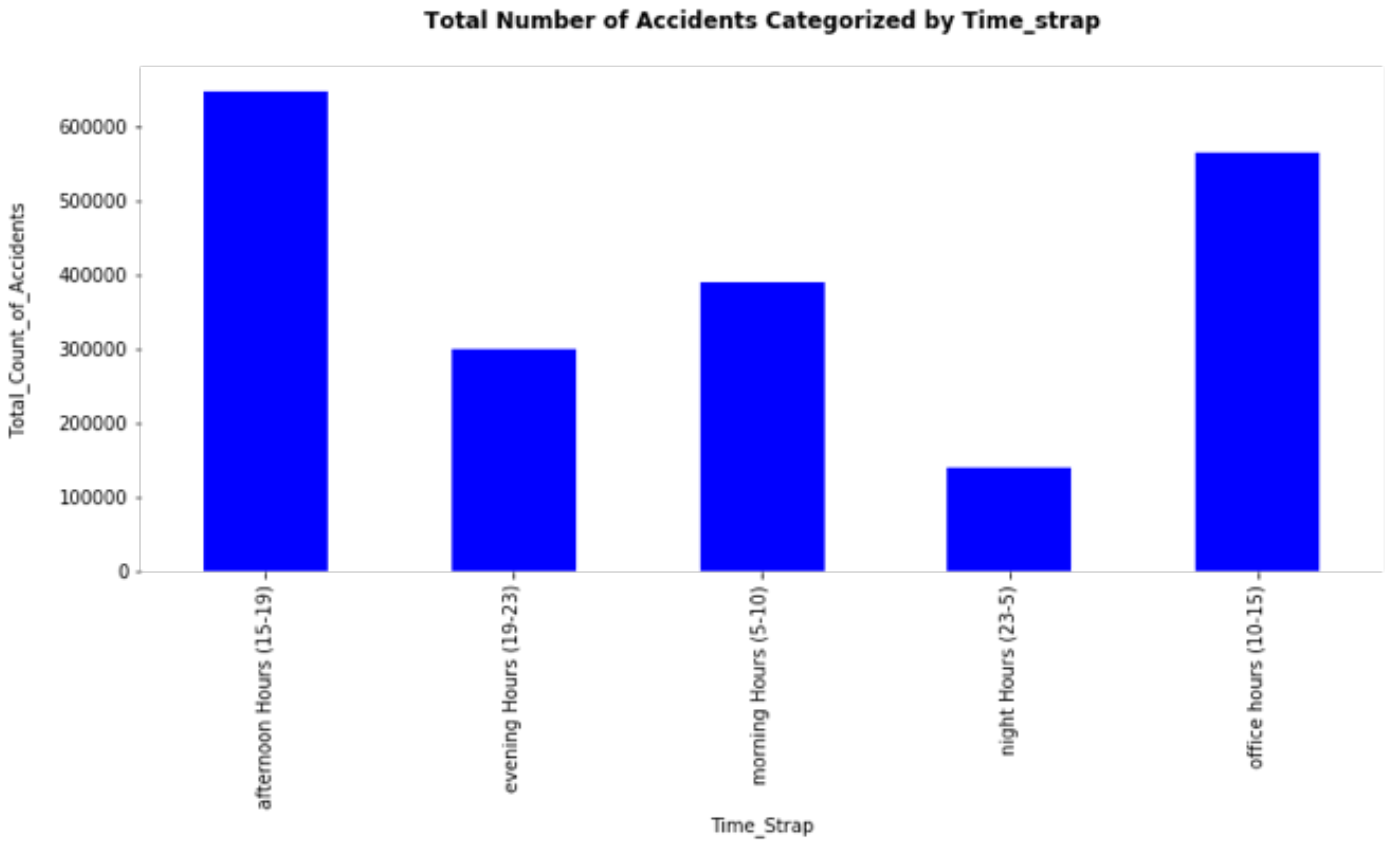
The old time and Hour column is no longer needed and hence they are dropped.

In [13]:
```
accidents_info = accidents_info.drop(columns=['Time', 'Hour'])
```

Since the occurrence of the events are grouped based on the time of the event happened, the number of events can be categorised based on the time strap as shown,

In [17]:
```
accidents_info.groupby('Time_Strap').size().plot(kind='bar', color='Blue', figsize=(12,5), grid
plt.xlabel('Time_Strap',rotation='horizontal')
plt.ylabel('Total_Count_of_Accidents\n')
plt.title('\nTotal Number of Accidents Categorized by Time_strap\n', fontweight='bold')
sns.despine(top=True, right=True, left=True, bottom=True);
```

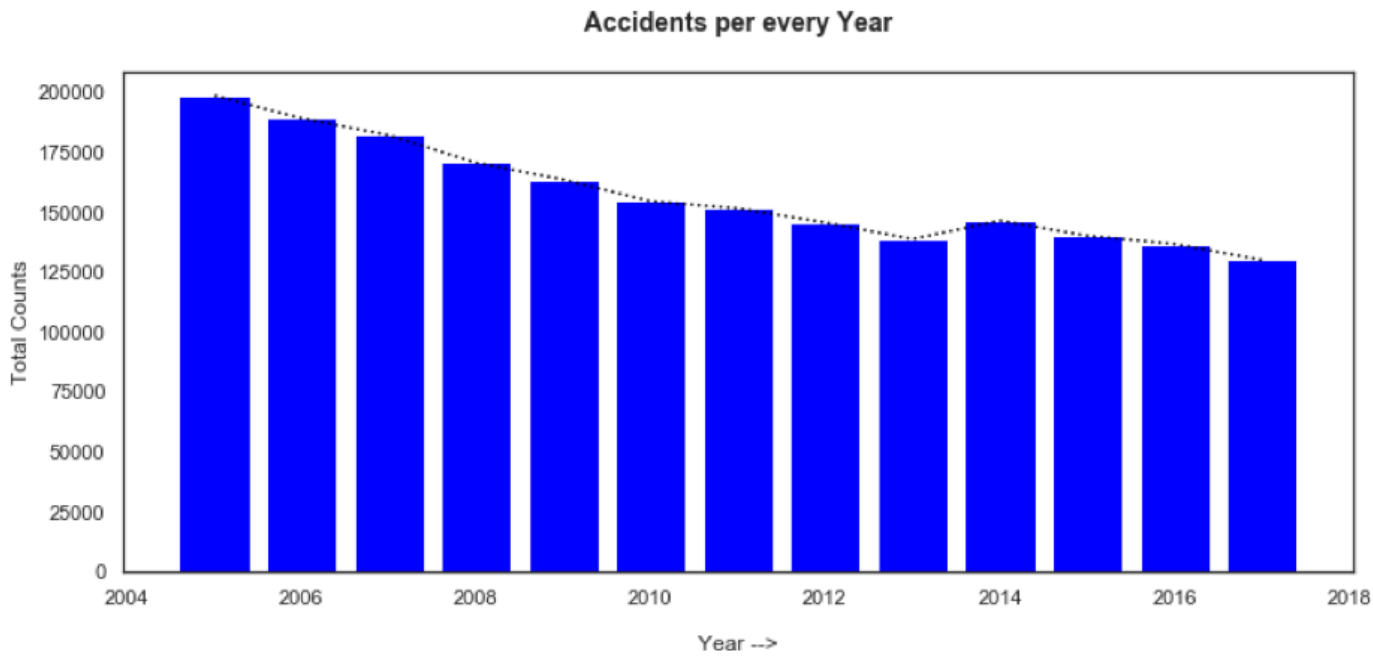
The below Graph shows that most of the accidents occurred are during office hours or afternoon hours respectively. This shows that most of them are prone to occur during regular hours than late nights,



With the Given information the Total number of accidents that happened in every year is considered and from which whether the accident count over these years are related and considered for analysis. The code and bar chart - illustration of Accidents per every year is as follows,

In [42]:
```
yearly_count = accidents_info['Date'].dt.year.value_counts().sort_index(ascending=False)
sns.set_style('white')
fig, ax = plt.subplots(figsize=(12,5))
ax.bar(yearly_count.index, yearly_count.values, color='blue')
ax.plot(yearly_count, linestyle=':', color='black')
ax.set_title('\nAccidents per every Year\n', fontsize=14, fontweight='bold')
ax.set(xlabel='\nYear -->')
ax.set(ylabel='\nTotal Counts')
```

Out[42]:
```
[Text(0, 0.5, '\nTotal Counts')]
```



From the above illustration , its clear that the Accidents from 2005 are considerably decreased due to all the active measures and safety precautions taken during each year. This shows one more aspect that the accident will be reduced every year from now on .

In [21]:
```
ax = sns.countplot(x = merged_df.Accident_Severity ,palette="Set2")
sns.set(font_scale=1)
ax.set_xlabel('Severity of the Accident ')
ax.set_ylabel('Total No of accidents ')
ax.set_title('Severity classification of Accidents',fontweight='bold' )
fig = plt.gcf()
fig.set_size_inches(8,4)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(100*p.get_height()/len(merged_df.Accident_Severity)), (p.get_x()+ 0.3, p.get_height()+
```

Both the Accident_information and Vehicle information are then merged using the common attribute Accident_index as shown,

```
In [15]: merged_df = pd.merge(vehicles_info, accidents_info, how = 'inner', on = 'Accident_Index')
```

```
In [16]: merged_df.head()
```

Out[16]:

	Accident_Index	Age_Band_of_Driver	Age_of_Vehicle	Driver_Home_Area_Type	Driver_IMD_Decile	Engine_Capacity_.CC.	Hit_O
0	200501BS00002	36 - 45	3.0	Data missing or out of range	NaN	8268.0	
1	200501BS00003	26 - 35	5.0	Urban area	3.0	8300.0	
2	200501BS00004	46 - 55	4.0	Urban area	1.0	1769.0	
3	200501BS00005	46 - 55	10.0	Data missing or out of range	NaN	85.0	
4	200501BS00006	46 - 55	1.0	Urban area	4.0	2976.0	

5 rows × 57 columns

This Merged data frame (merged_df) consists of 57 columns . Not all these attributes do actually contribute to the data set. Therefore considering the ones which does actually contribute to the predictions or analysis.

The df.describe() function gives the basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.

```
In [28]: df.describe()
```

Out[28]:

	Number_of_Casualties	Number_of_Vehicles	Speed_limit	Age_of_Vehicle	Engine_Capacity_.CC.
count	2.058262e+06	2.058262e+06	2.058197e+06	1.720336e+06	1.807970e+06
mean	1.449619e+00	2.124039e+00	3.975690e+01	7.143342e+00	2.028279e+03
std	1.033511e+00	9.641712e-01	1.460913e+01	4.728791e+00	1.914112e+03
min	1.000000e+00	1.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00
25%	1.000000e+00	2.000000e+00	3.000000e+01	3.000000e+00	1.299000e+03
50%	1.000000e+00	2.000000e+00	3.000000e+01	7.000000e+00	1.598000e+03
75%	2.000000e+00	2.000000e+00	5.000000e+01	1.000000e+01	1.997000e+03
max	9.300000e+01	6.700000e+01	7.000000e+01	1.110000e+02	9.600000e+04

The proportion of missing values found in accident_information and Vehicle_information are found to be 0.495 and 0.938% respectively. Hence ignoring those values, further calculations are made.

In [29]:
```
print('Missing Values % in Accidents_info :',
      round(accidents_info.isna().sum().sum()/len(accidents_info),3), '%')
```

```
Missing Values % in Accidents_info : 0.495 %
```

In [30]:
```
print('Missing Values % in vehicles_info :',
      round(vehicles_info .isna().sum().sum()/len(vehicles_info),3), '%')
```

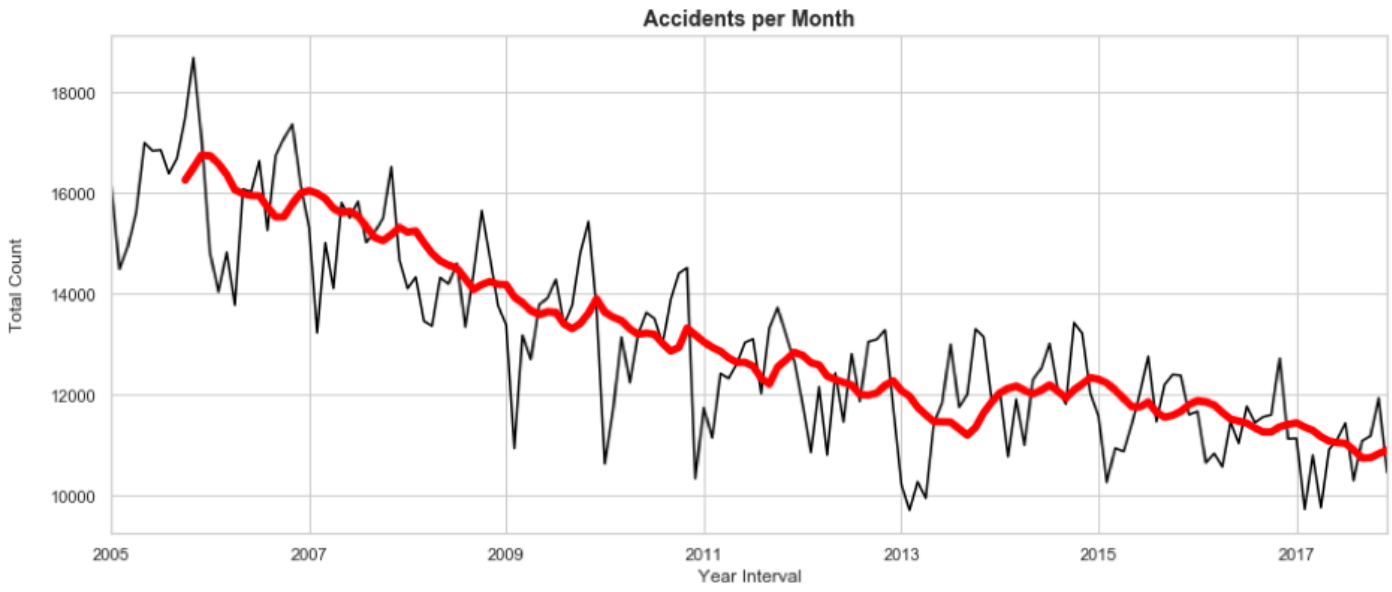
```
Missing Values % in vehicles_info : 0.938 %
```

OBSERVATIONS

A graph has been computed with accidents happened per month with subplot of accidents average over every 10 months. This gives us a clear idea of what linearity of data occurrence has

In [31]:
```
sns.set_style('whitegrid')
fig, ax = plt.subplots(figsize=(15,6))
accidents_info.set_index('Date').resample('M').size().plot(label='Total Accidents per Month', color='black', ax=ax)
accidents_info.set_index('Date').resample('M').size().rolling(window=10).mean()\
                          .plot(color='red', linewidth=5, label='Average for every 10 months', ax=ax)
ax.set_title('Accidents per Month', fontsize=14, fontweight='bold')
ax.set(ylabel='Total Count\n')
ax.set(xlabel='Year Interval\n')
```

Out[31]:
```
[Text(0.5, 0, 'Year Interval\n')]
```



• It has been found that most of the drivers involving in the accidents are male covering almost 67% of entire accident counts. Females where recorded to be 29% and few of the accidents have not recorded any information with respect to the Driver_age.

```
In [32]: vehicles_info.Sex_of_Driver.value_counts(normalize=True)
```

```
Out[32]: Male                0.674296
         Female              0.290742
         Not known           0.034931
```

By dropping those missing values and some which are not in range, Driver_details has been created with 'Age_Band_of_Driver', 'Sex_of_Driver', 'Count' as shown as ,

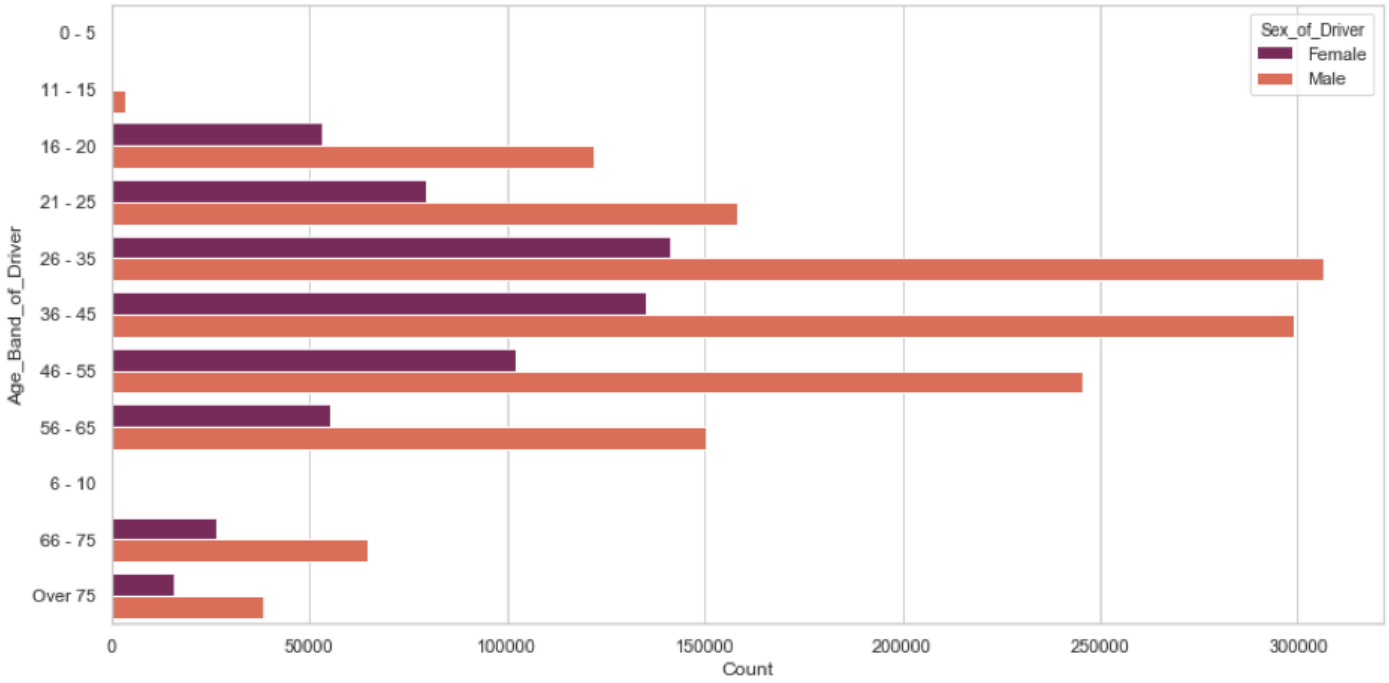
```
In [33]: driver_details = vehicles_info.groupby(['Age_Band_of_Driver', 'Sex_of_Driver']).size().reset_index()
```

```
In [34]: driver_details.drop(driver_details[(driver_details['Age_Band_of_Driver'] == 'Data missing or out of range') | \
                        (driver_details['Sex_of_Driver'] == 'Not known') | \
                        (driver_details['Sex_of_Driver'] == 'Data missing or out of range')]\
                        .index, axis=0, inplace=True)
```

```
In [35]: driver_details.columns = ['Age_Band_of_Driver', 'Sex_of_Driver', 'Count']
```

Thus a barplot has been generated to describe the majority of the accidents happended by , and their count as shown,

```
ax = plt.subplots(figsize=(14, 7))
sns.barplot(y='Age_Band_of_Driver', x='Count', hue='Sex_of_Driver', data=driver_details, palette='rocket')
ax.set_title('\nDrivers by Age and Sex\n', fontsize=14, fontweight='bold')
ax.set(xlabel='Accident Counts', ylabel='Age Band of Driver')
ax.legend(bbox_to_anchor=(1.1, 1.), borderaxespad=0., frameon=False)
```



The Below image has data being recorded for drivers with age group 11-15 which is technically not possible . Therefore these outliers are to be exempted.

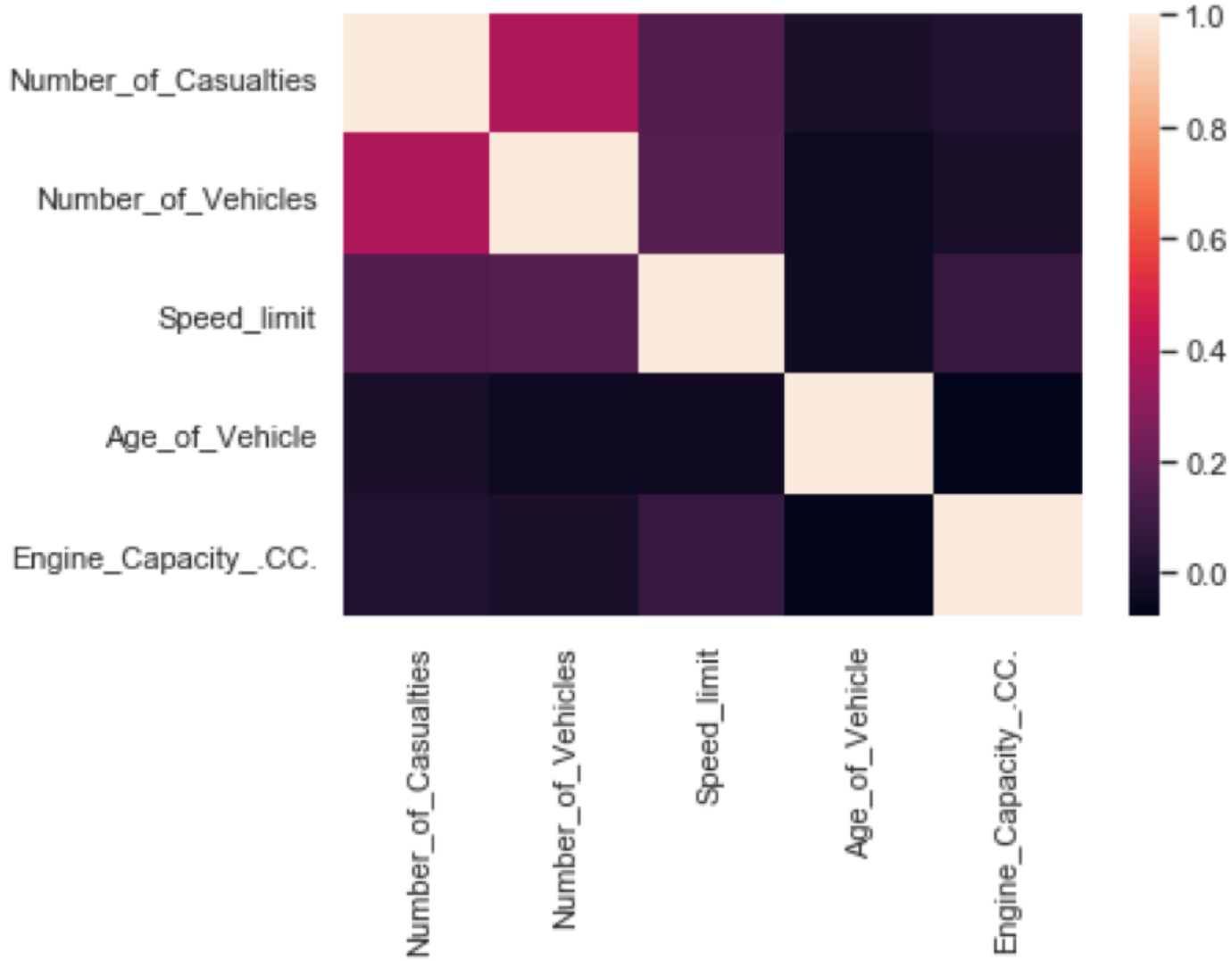
• A heatmap has been generated based on the correlation values of Number_of_Casualties, Number_of_Vehicles, Speed_limit, Age_of_Vehicle and Engine*Capacity*.CC.

In [44]:
```
df.corr()
```

Out[44]:

	Number_of_Casualties	Number_of_Vehicles	Speed_limit	Age_of_Vehicle	Engine_Capacity_.CC.
Number_of_Casualties	1.000000	0.382233	0.148501	-0.004052	0.013740
Number_of_Vehicles	0.382233	1.000000	0.161666	-0.041110	-0.004579
Speed_limit	0.148501	0.161666	1.000000	-0.033289	0.078281
Age_of_Vehicle	-0.004052	-0.041110	-0.033289	1.000000	-0.077765
Engine_Capacity_.CC.	0.013740	-0.004579	0.078281	-0.077765	1.000000

The Correlation values of these columns can be plotted for better visibility and shown as below,



DATA MODELING

The next part of this project is to predict the accident severity , whether the accident happened is slight or severe. To discover this , several models are built to train our system and test the model with another test set for its accuracy and results.

The X and Y are being set as shown ,

In [47]:
```
X= df[['Accident_Index', '1st_Road_Class', 'Day_of_Week', 'Junction_Detail',
       'Light_Conditions', 'Number_of_Casualties', 'Number_of_Vehicles',
       'Road_Surface_Conditions', 'Road_Type', 'Special_Conditions_at_Site',
       'Speed_limit', 'Time_Strap', 'Urban_or_Rural_Area',
       'Weather_Conditions', 'Age_Band_of_Driver', 'Age_of_Vehicle',
       'Hit_Object_in_Carriageway', 'Hit_Object_off_Carriageway', 'make',
       'Engine_Capacity_.CC.', 'Sex_of_Driver', 'Skidding_and_Overturning',
       'Vehicle_Manoeuvre', 'Vehicle_Type',]]
```

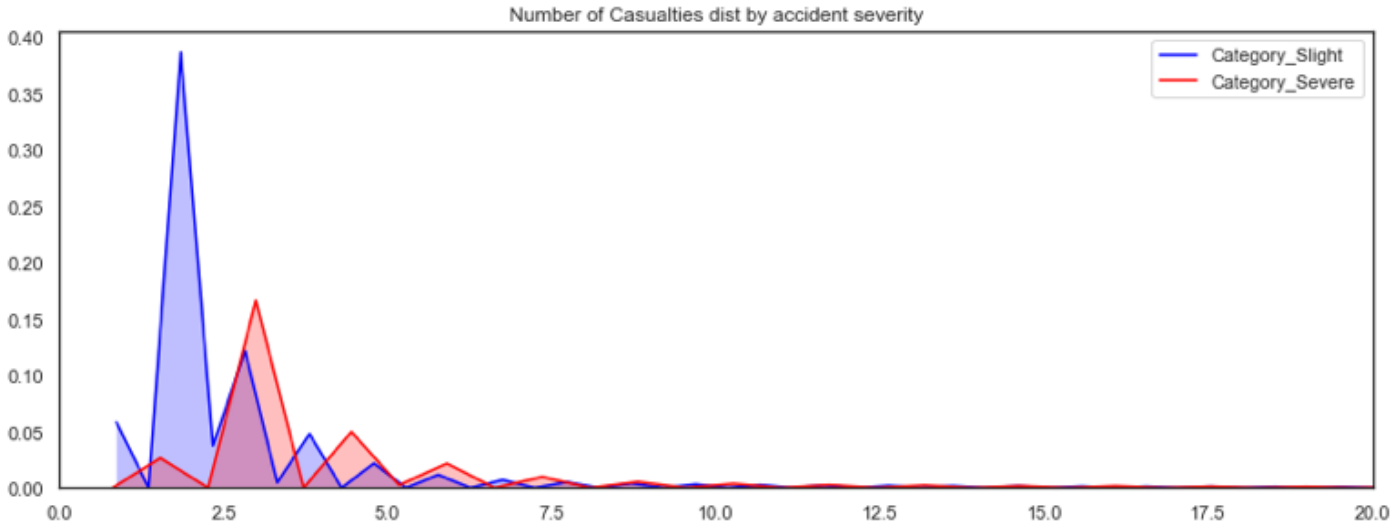
In [48]:
```
y = df['Accident_Severity']
```

Since the Accident_Severity has values slight, severe and fatal, its hard to compute. Hence replacing fatal with serious as fatal comes under serious severity which requires immediate action. Thus replacing fatal with serious and normalising the factor can be given as ,

In [49]:
```
df['Accident_Severity'] = df['Accident_Severity'].replace(['Serious', 'Fatal'], 'Serious or Fatal')
df = pd.get_dummies(df, columns=['Accident_Severity'])
df = df.drop('Accident_Severity_Serious or Fatal', axis=1)
df.Accident_Severity_Slight.value_counts(normalize=True)
```

Thus normalising the values and plotting them with number of causalities distributed over, the graph illustrates the distribution of number of causalities with accident_severity _slight or Severe.

In [52]:
```
plt.figure(figsize=(14,5))
acc_slight = df.Accident_Severity_Slight == 1
acc_severe = df.Accident_Severity_Slight == 0
sns.kdeplot(df.Number_of_Casualties[acc_slight],shade=True,color='Blue', label='Category_Slight').set_xlim(0,20)
sns.kdeplot(df.Number_of_Casualties[acc_severe],shade=True,color='Red', label='Category_Severe').set_xlim(0,20)

plt.title('Number of Casualties dist by accident severity')
plt.show()
```



PREDICTIVE ANALYSIS

Linear Regression

Linear Regression mainly aims to build the relationship between two variables by fitting a linear equation to the observed data. Out of those two variables one is considered to be an explanatory variable and other is considered to be a dependent variable. The training and testing of those data and its outcomes are explained further as shown,

```
In [45]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.4, random_state=101)
         from sklearn.linear_model import LinearRegression
         lm = LinearRegression()
         X_train.fillna(X_train.mean(), inplace=True)
```

Training the model to facilitate the test data and get predictions out of it.

```
In [46]: lm.fit(X_train, y_train)

Out[46]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)

In [47]: print(lm.intercept_)

         0.9407455679067871

In [48]: lm.coef_

Out[48]: array([-3.37454688e-02,  2.52886605e-02, -1.89415178e-03, -1.58353156e-03,
                -4.44271916e-07])
```

The coefficients of those variables and their score can be found using linear regression as shown below,

In [49]:
```
cdf= pd.DataFrame(lm.coef_,X.columns,columns=['Coeff'])
```

In [50]:
```
cdf
```

Out[50]:

	Coeff
Number_of_Casualties	-3.374547e-02
Number_of_Vehicles	2.528866e-02
Speed_limit	-1.894152e-03
Age_of_Vehicle	-1.583532e-03
Engine_Capacity_.CC.	-4.442719e-07

Logistic Regression

Logistic regression is conducted when the dependent variable is binary. This is a kind of predictive analysis which is used to describe the data and explain the relationship between one dependent variable with the binary variable.

In [61]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.3, random_state=101)
from sklearn.linear_model import LogisticRegression
```

In [64]:
```
logmodel=LogisticRegression()
X_train.fillna(X_train.mean(), inplace=True)
X_test = X_test.fillna(X_train.mean())

X_test.fillna(X_train.mean(), inplace=True)
```

Thus the Accident Severity is categorised by Slight or Severe with 0s and 1s. The classification report can be shown as below,

In [70]:
```
predictions = logmodel.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.45      0.00      0.01     87846
           1       0.86      1.00      0.92    529633

   micro avg       0.86      0.86      0.86    617479
   macro avg       0.66      0.50      0.46    617479
weighted avg       0.80      0.86      0.79    617479
```

In [65]:
```
logmodel.fit(X_train,y_train)
```

```
/Users/vishnumohan/anaconda3/lib/python3.7/site-packages/sklearn/linear_model
solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this w
  FutureWarning)
```

Out[65]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [80]:
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

In [73]:
```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

In [81]:
```
probs = logmodel.predict_proba(X_test)
probs = probs[:, 1]
auc = roc_auc_score(y_test, probs)
print('AUC: %.2f' % auc)
```

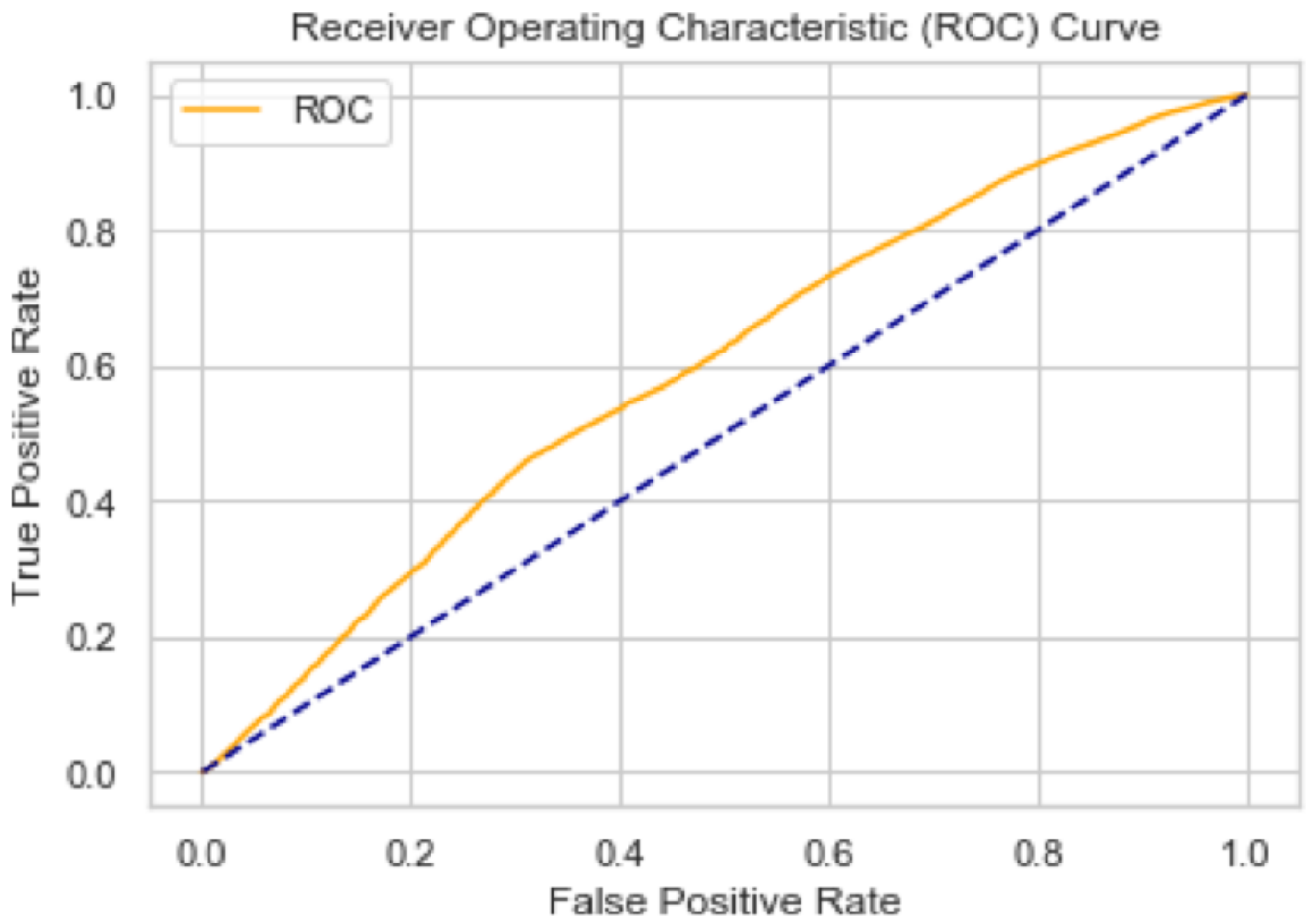
```
AUC: 0.60
```

In [82]:
```
fpr, tpr, thresholds = roc_curve(y_test, probs)
```

In [83]:
```
plot_roc_curve(fpr, tpr)
```

ROC Curve for Logistic Regression:

The Roc curve for Logistic Regression can be plotted as shown below,



Decision Tree Classifier

This Model uses a tree like graph or model of decisions and compute the results. They are also called as powerful machine learning model which is capable of providing high accuracy when the data is highly interpretable. Firstly determine the best features which actually contributes to the output and split the data into two train and test.

In [56]:
```
X = X_feat
y = df['Accident_Severity_Slight']
X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.3, random_state=101)
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
```

The following Context shows the steps involved in training and testing the model and predicting the results.

Split and Train the model ,

In [60]:
```
dtree.fit(X_train,y_train)
```

Out[60]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [61]:
```
predictions = dtree.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
```

The prediction results of decision tree can be given as,

In [63]:
```
predictions = dtree.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,predictions))
print('\n')
print(classification_report(y_test,predictions))
```

```
[[  9090  78756]
 [ 22649 506984]]


              precision    recall  f1-score   support

           0       0.29      0.10      0.15     87846
           1       0.87      0.96      0.91    529633

   micro avg       0.84      0.84      0.84    617479
   macro avg       0.58      0.53      0.53    617479
weighted avg       0.78      0.84      0.80    617479
```

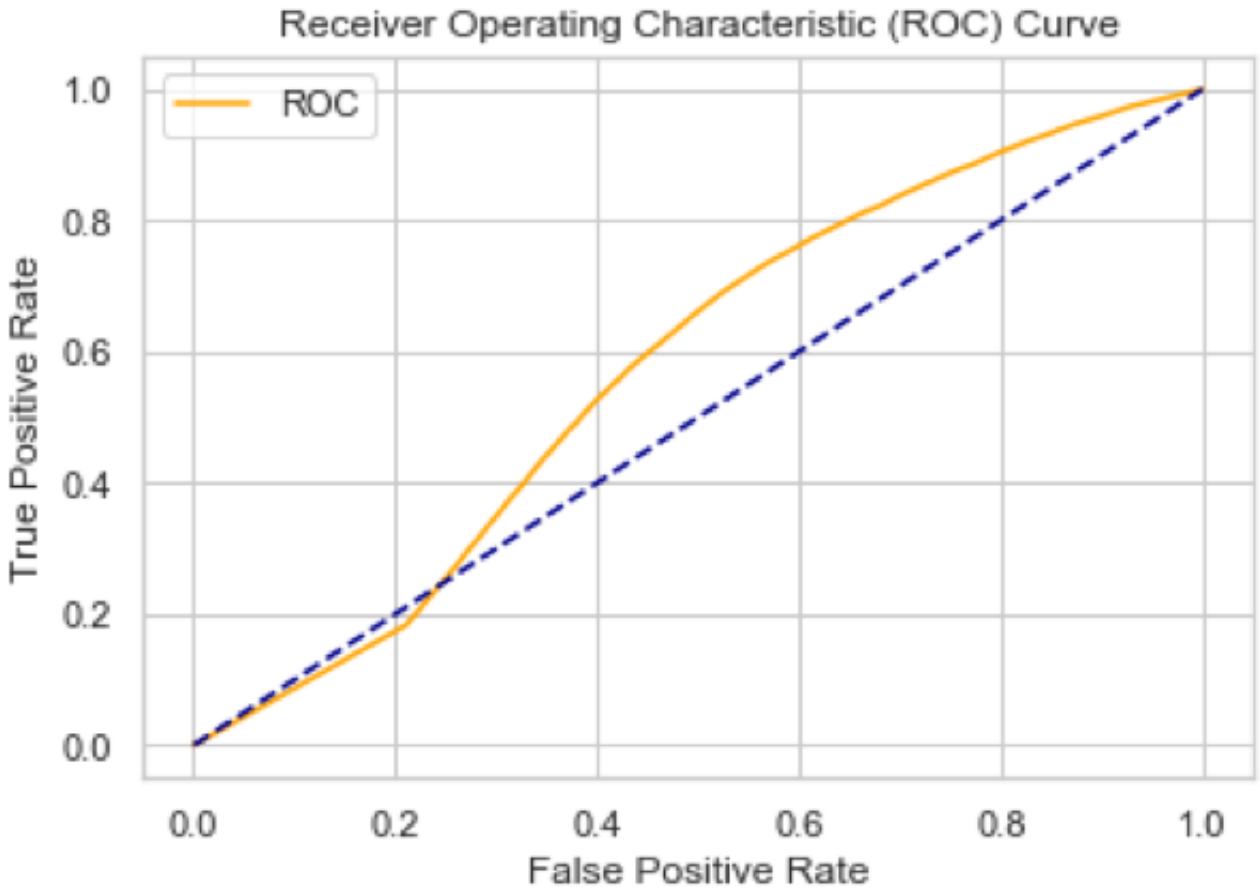
The ROC curve for Decision tree algorithm is as shown as ,

In [65]:
```
probs = dtree.predict_proba(X_test)
probs = probs[:, 1]
auc = roc_auc_score(y_test, probs)
print('AUC: %.2f' % auc)
```

```
AUC: 0.58
```

In [66]:
```
fpr, tpr, thresholds = roc_curve(y_test, probs)
```

In [67]:
```
plot_roc_curve(fpr, tpr)
```



Random Forest Classifier

Random Forest - as the name implies it mainly consists of a large number of individual decision trees that operate as ensemble. Each tree computes and provides a prediction , further each tree in the random forest spits out a prediction and one with most votes becomes the model's prediction. Thus the Computational results of Random forest classifier is shown as below,

```
In [77]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.pipeline import Pipeline, FeatureUnion
         from sklearn.preprocessing import MinMaxScaler, FunctionTransformer, OneHotEncoder, KBinsDiscretizer, MaxAbsScaler

In [78]: rfc = Pipeline([("Min_Max_Transformer", MaxAbsScaler()),("Clf",RandomForestClassifier(n_estimators=100, n_jobs=3))])

In [79]: rfc.fit(X_train, y_train)

Out[79]: Pipeline(memory=None,
              steps=[('Min_Max_Transformer', MaxAbsScaler(copy=True)), ('Clf', RandomForestClassifier(bootstrap=True, class_we
         ight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=3,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False))])
```

The prediction and accuracy results are as shown as ,

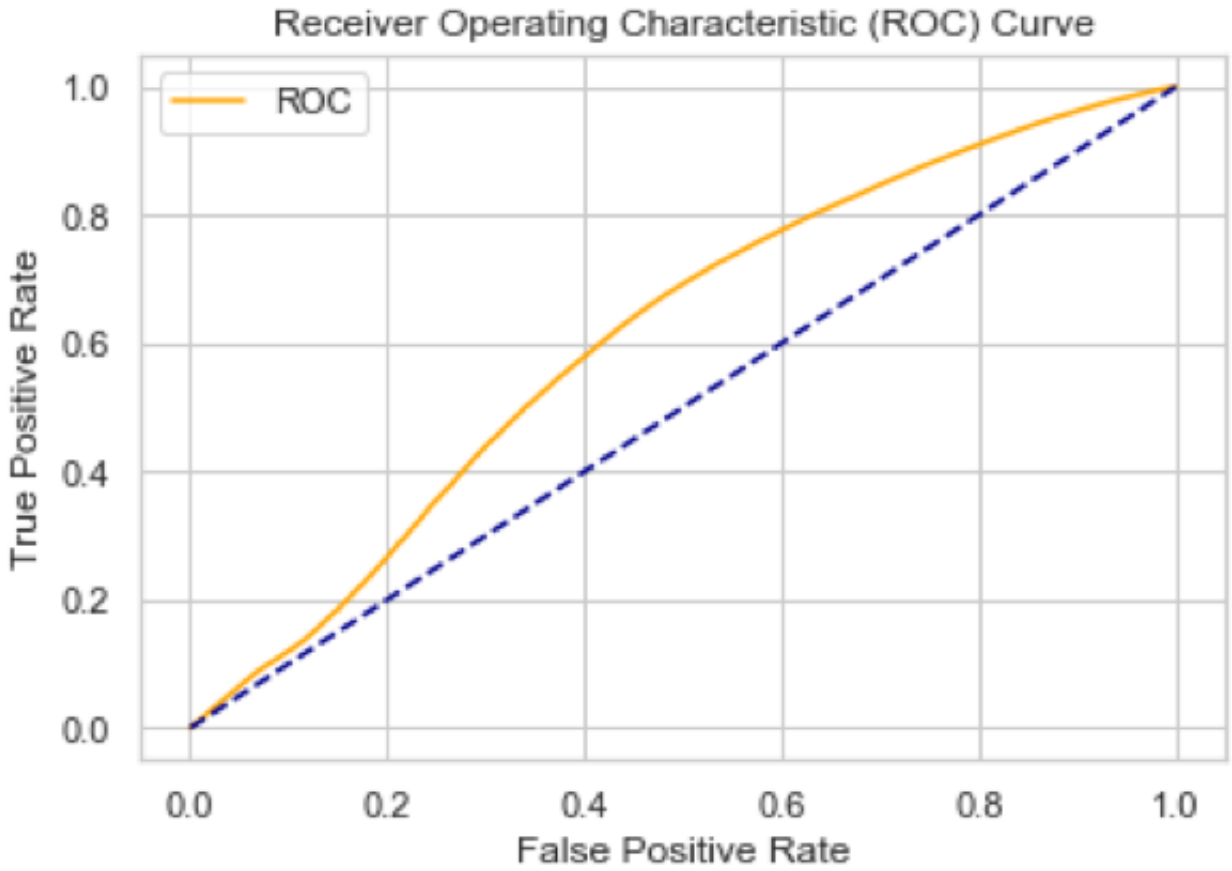
```
In [80]: rfc.pred = rfc.predict(X_test)

In [81]: print(confusion_matrix(y_test,rfc.pred))
         print('\n')
         print(classification_report(y_test,rfc.pred))
```

```
[[  6501  81345]
 [ 13869 515764]]


              precision    recall  f1-score   support

           0       0.32      0.07      0.12     87846
           1       0.86      0.97      0.92    529633

   micro avg       0.85      0.85      0.85    617479
   macro avg       0.59      0.52      0.52    617479
weighted avg       0.79      0.85      0.80    617479
```

The results and ROC curve of the Random forest classifier is as shown as,

In [81]:
```
probs = rfc.predict_proba(X_test)
probs = probs[:, 1]
auc = roc_auc_score(y_test, probs)
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.61
```



TIME SERIES ANALYSIS - FORECAST

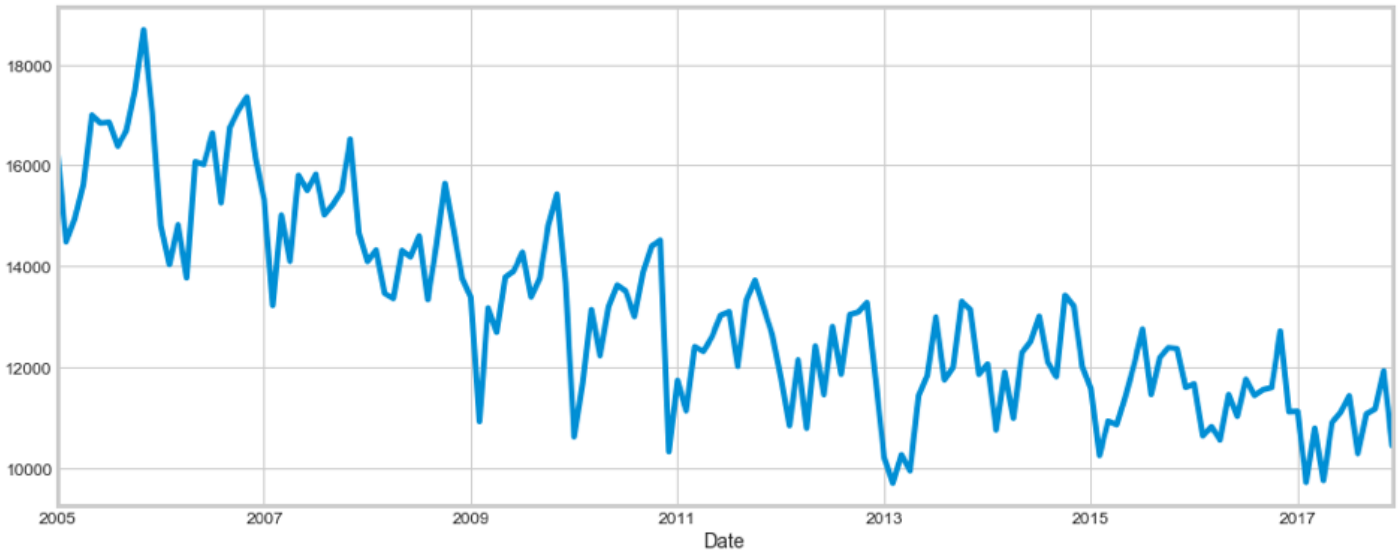
Time Series Analysis generally comprise of methods and techniques to analyse time series data with an aim of predicting and extracting meaningful statistics and other characteristics of data. Time series Forecasting model is used to predict the future values or expectations with previously observed values. These are mostly deployed to the non-stationary data like , weather , Stock price, retail sales , economic situations , accident counts etc..

There are several Categories in the Accident_info data , we consider the date for accounting the number of accidents happened in a year through time series analysis and forecasting.

In [94]:
```
y=accidents_info.set_index('Date').resample('M').size()
y.plot(figsize=(15, 6))
plt.show()
```

Since All the default attributes and information are already loaded , moving onto visualising the no of accidents happened in the past years.

Some varying patterns has been observed after plotting the data, The time-series analysis mostly has a seasoning pattern and it keeps on varying throughout the year as shown below, .



Thus this can be decomposed again to three distinct components,

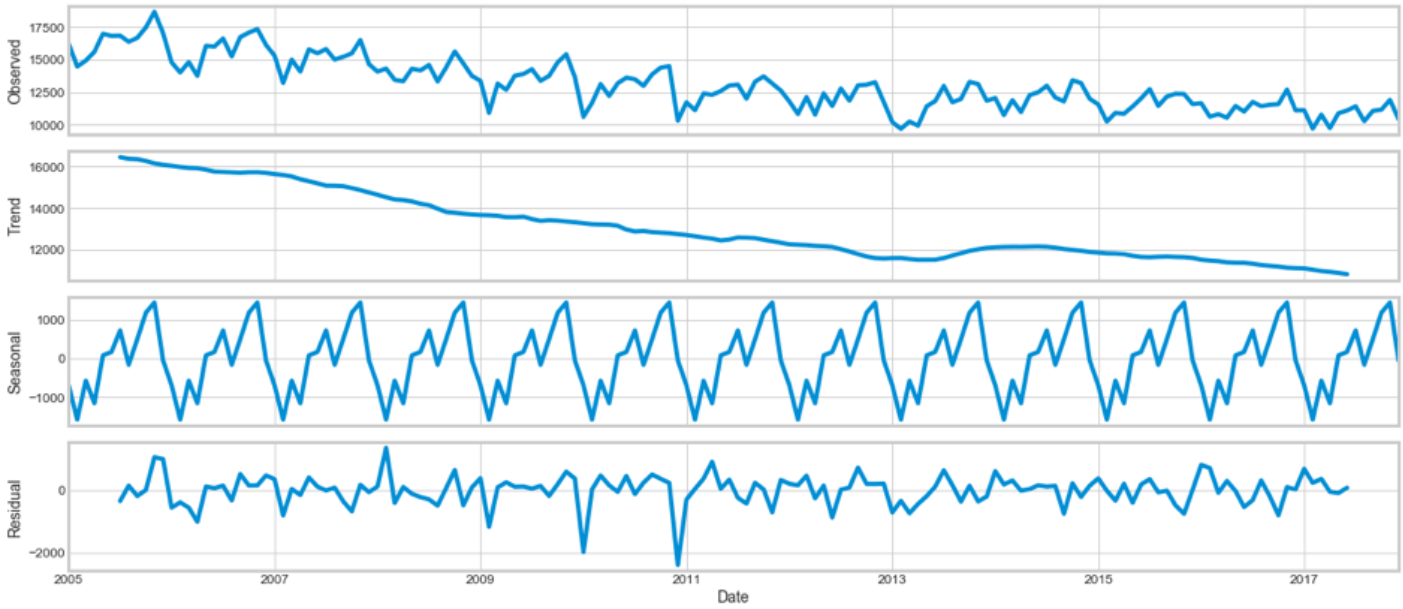
• Trend

• Seasonality

• Noise

In [81]:
```
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```

The Visualisation with the decomposed components can be shown as ,



Time Series Forecasting _ ARIMA:

Implying time series forecasting known as ARIMA , will be the best option to distinguish the time series analysis of data. The models are explained with notation ARIMA(p, d, q)as it symbolises seasonality, Trend, and noise in data.

In [82]:
```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

The next steps would be parameter selection to find the number of accidents using time series model. To facilitate this , the optimal parameters which actually contribute to the performance of the model are observed.

In [83]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

The generated output shows, ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:2027.0669447423975 and this should be considered as the optimal value.

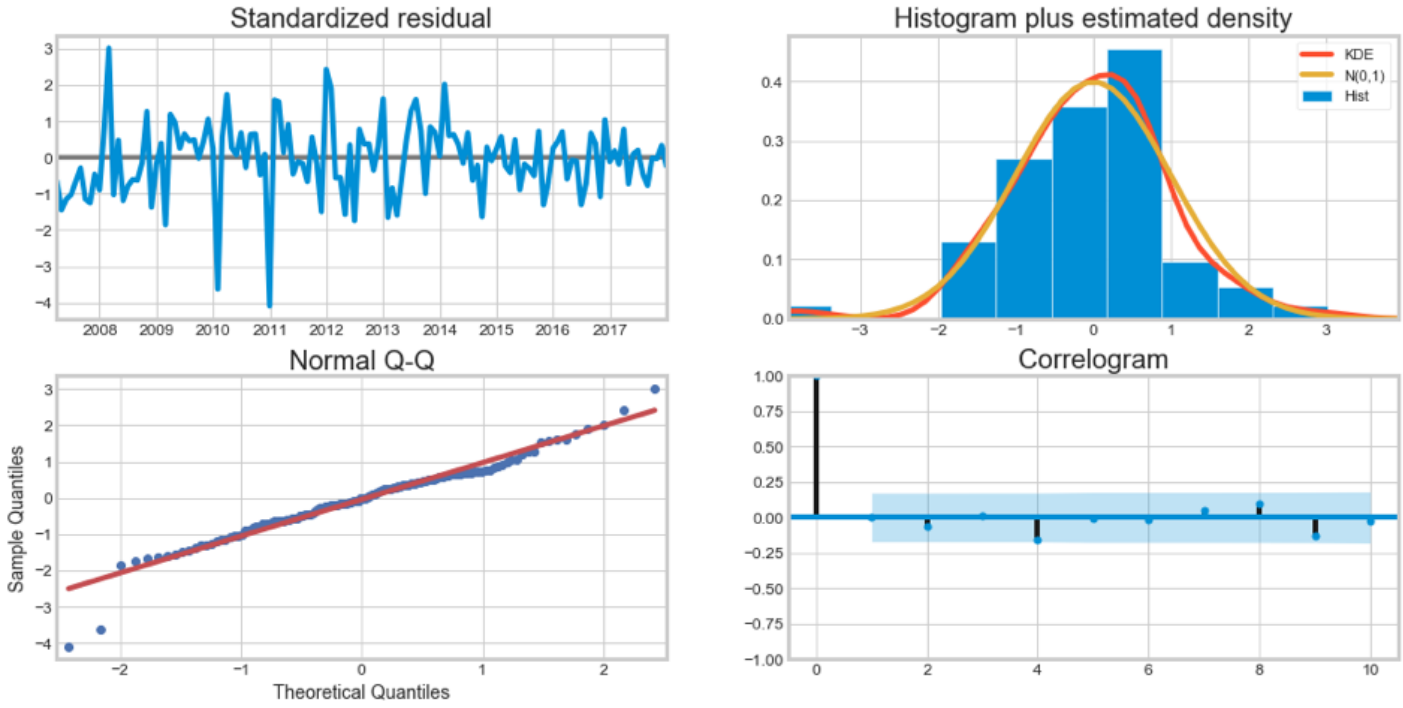
The next procedure would be fitting the Arima Model as shown in the below figure,

In [84]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0488      0.116     -0.422      0.673      -0.275       0.178
ma.L1         -0.6527      0.087     -7.476      0.000      -0.824      -0.482
ar.S.L12      -0.5295      0.062     -8.569      0.000      -0.651      -0.408
sigma2       4.67e+05   4.23e+04     11.032      0.000    3.84e+05     5.5e+05
==============================================================================
```

Check whether the model is running as expected by plotting the diagnostics. This will showcase us in-case of any unusual behaviour experienced.

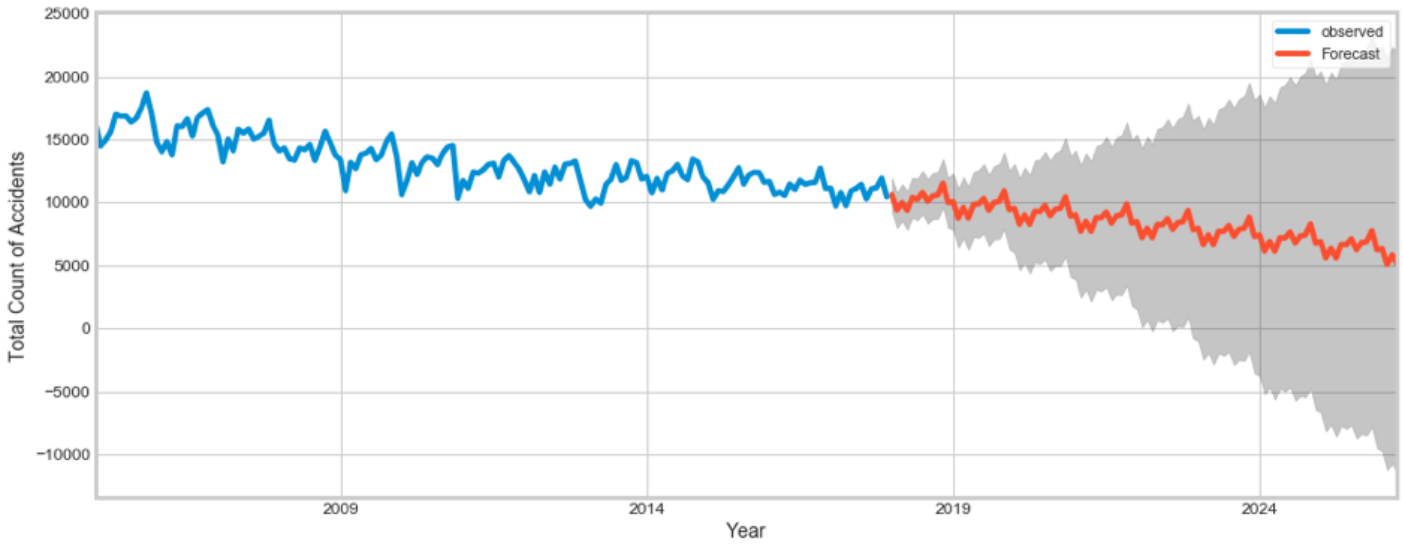
In [85]:
```
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



To predict the forecast we have the our model ready to go , as we forecast the accident count further into future as shown ,

In [93]:
```
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='observed', figsize=(15, 6))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Year')
ax.set_ylabel('Total Count of Accidents')
plt.legend()
plt.show()
```

The below graph shows the forecast of accident count , developed from our observed model.



Thus Time Series Analysis results shows that number of accidents forecasted further to the future years are decreasing .Thus with proper actions, remedies and precautions taken from the government and other stakeholder, the number of accidents count, the severity and the number of causalities will considerably be reduced.

CONCLUSION

The main aim of this project is to provide Predictive analysis of Accidents and its severity for the forthcoming years. Various machine learning algorithms have been implemented to predict the models and analyse the severity so that the results can be beneficial to the stakeholders to take necessary precautions and actions to avoid them in the future . Further The Time series analysis implementation has helped to have clear view of the forecast that how the severity of the accidents would be , places needs more attention , remedial action that has to be taken into account . Thus these analysis techniques can be used to avoid accidents and help passengers or causalities in a timely manner and can help government to helps the victims without any delay and make better decisions in road investments and traffic rules.