##

TRUCK DELIVERY Modeling by Implementing LOGISTIC REGRESSION

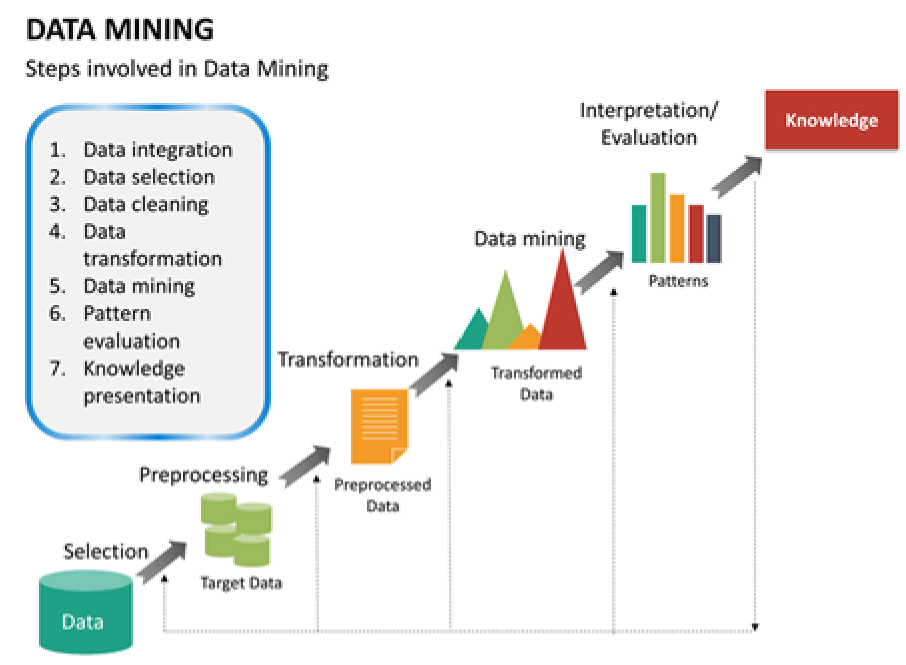
##

By: Sara Khosravi

Instructor: MS.Giti Saikia

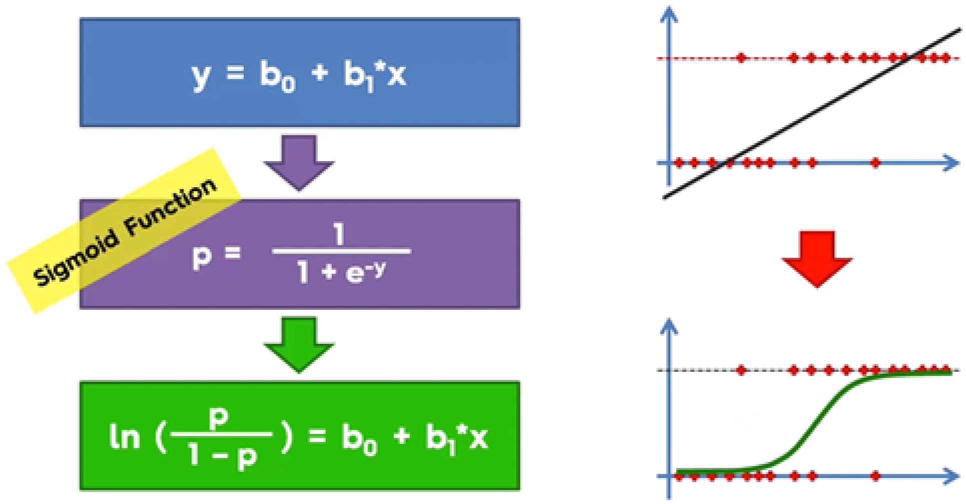
##

***DATA MINING COLLECT DATA AND MAKE DECISION



Logistic Regression Overview

- Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression. Logistic regression is fast and relatively uncomplicated, and it's convenient for you to interpret the results. Although it's essentially a method for binary classification, it can also be applied to multiclass problems.
- Independent variables, also called inputs or predictors, don't depend on other features of interest (or at least you assume so for the purpose of the analysis).
- Dependent variables, also called outputs or responses, depend on the independent variables.

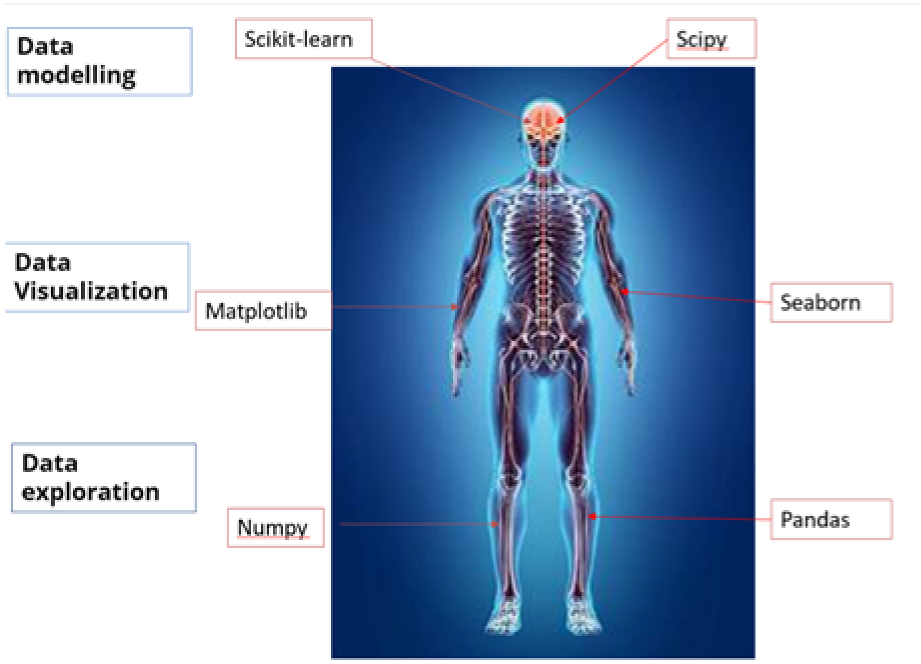


In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries inst
# For example, here's several helpful packages to load

import warnings
warnings.filterwarnings('ignore')

import numpy as np # linear algebra
#Import pandas for the data-structures
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

import pandas_profiling

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will l

import os
os.chdir(r'C:\Sara\Data SCIENCE\DataMining\Project')
os.getcwd()

```

Out[1]:

```
'C:\\Sara\\Data SCIENCE\\DataMining\\Project'
```



What are the types of data analysis in Python?

Data can be in any of the popular formats - CSV, TXT, XLS/XLSX (Excel), sas7bdat (SAS), Stata, Rdata (R) etc. Loading data in python environment is the most initial step of analyzing data. Date Type variable in consistent date format. pandas is a powerful data analysis package. It makes data exploration and manipulation easy.

DATA COLLECTION

DATA FROM GEOTAB COMPANY

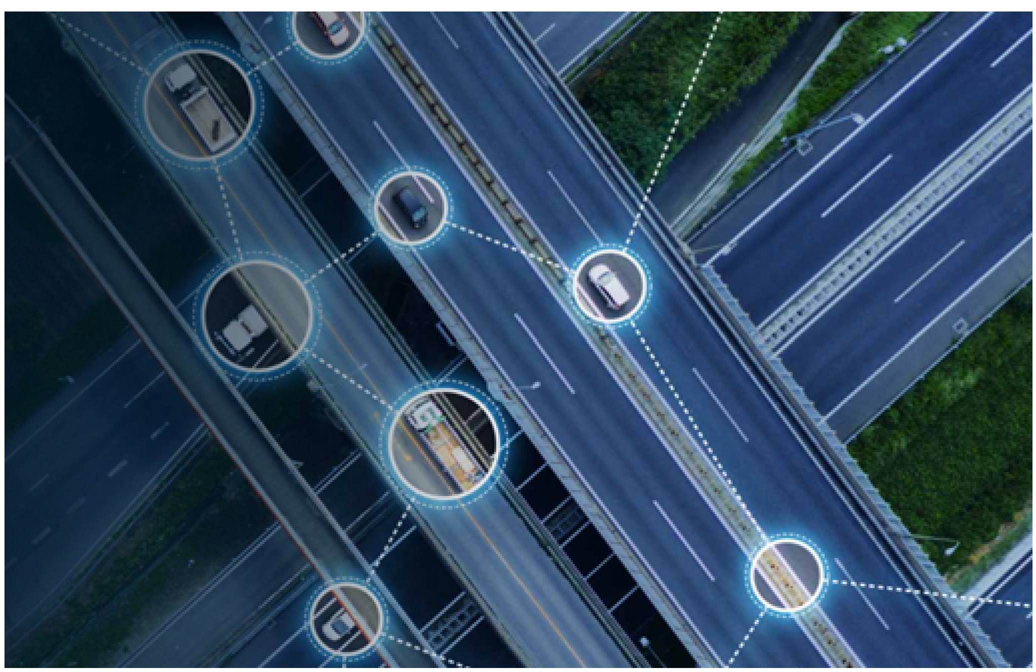
[https://data.geotab.com/intelligence-data (https://data.geotab.com/intelligence-data)](https://data.geotab.com/intelligence-data)

- ELD's are electronic devices that automatically record a driver's hours of service. Such technology allows for accurate recordkeeping of electronic logs. An ELD device tracks a truck's engine and captures data on its movement, whether the truck is in motion or idling.
- BUT this dataset collected by GPS

- As a Logistics company, it is necessary to provide an effective trip experience to the customer in an optimized cost.
- To provide an effective service we need to identify the parameters that impact the on-time arrival of the truck. With the pattern formed we need to formulate the data points that would help reduce the trip cost

```
About this file
Dataset Description:
GpsProvider - Vendor who provides GPS
```

```
BookingID - Unique Identification for a trip
Market/Regular - Type of trip. Regular - Vendors with whom we will have contract. Market - Vendor with whom we will not have contract
BookingIDDate - Date when booking was created vehicleno - Truck Number
OriginLocation - Trip start place DestinationLocation - Trip end place
Orglatlon - Latitude/Longitude of start place
Deslatlon - Latitude/Longitude of end place
DataPingtime - Time when we receive GPS ping
PlannedETA - Planned Estimated Time of Arrival CurrentLocation - Live location
DestinationLocation - Repeat of destination location
actualeta - Time when the truck arrived Currlat - current latitude - changes each time when we receive GPS ping
Currlon - current longitude - changes each time when we receive GPS ping ontime - If the truck arrived on time - calculated based on Planned and Actual ETA delay - If the truck arrived with a delay - calculated based on Planned and Actual ETA OriginLocationCode - Origin code
DestinationLocationCode - Destination code tripstartdate - Date/Time when trip started tripenddate Date/Time when trip ended - based on documentation (cant be considered for calculating delay)\ TRANSPORTATIONDISTANCEINKM - Total KM of travel
vehicleType - Type of Truck
Minimumkmstobecoveredinaday - Minimum KM the driver needs to cover in a day DriverName - Driver details
Driver_MobileNo - Driver details
customerID - Customer details
customerNameCode - Customer details
supplierID - Supplier - Who provides the vehicle
supplierNameCode - Supplier - Who provides the vehicle
```



In [2]:

```python
#import the dataset
data=pd.read_excel(r'C:\Sara\Data SCIENCE\DataMining\Project\Delivery truck
data.shape
```

Out[2]: (6880, 32)

DataFrame - head() function

Pandas DataFrame head () Method in Python By Ankit Lathiya Last updated May 26, 2020 Pandas DataFrame head () method returns top n rows of a DataFrame or Series where n is a user input value. The head () function is used to get the first n rows. The head () function is used to get the first n rows. It is useful for quickly testing if your object has the right type of data in it. For negative values of n, the head () function returns all rows except the last n rows, equivalent to df [:-n]. The head () method in python contains only one parameter, which is n.

In [3]:

```python
data.head(5)
```

0	CONSENT TRACK	MVCV0000927/082021	Market	2020-08-17 14:59:01.000	KA590408	HUB,CH
1	VAMOSYS	VCV00014271/082021	Regular	2020-08-27 16:22:22.827	TN30BC5917	DAIMLER VEHICLE
2	CONSENT TRACK	VCV00014382/082021	Regular	2020-08-27 17:59:24.987	TN22AR2748	PONDY,PO
3	VAMOSYS	VCV00014743/082021	Regular	2020-08-28 00:48:24.503	TN28AQ0781	DAIMLER VEHICLE
4	VAMOSYS	VCV00014744/082021	Regular	2020-08-28 01:23:19.243	TN68F1722	PONDY,PO

5 rows × 32 columns

Looking at the data, it is seen that the data is an unstructured data.

DataFrame - tail() function

The tail() function is used to get the last n rows. This function returns last n rows from the object based on position. It is useful for quickly verifying data, for example, after sorting or appending rows. Syntax: DataFrame.tail(self, n=5) Parameters:

In [7]:
```
data.tail(5)
```

6875	JTECH	WDSBKTP42751	Regular	2019-03-27 17:25:33	KA219502	Ramamu Na Bangal Karnat
6876	JTECH	WDSBKTP43203	Regular	2019-03-31 15:02:34	KA01AE9163	Ramamu Na Bangal Karnat
6877	JTECH	WDSBKTP43021	Regular	2019-03-29 18:56:26	KA01AE9163	Mugab Bangalore Ru Karnat
6878	JTECH	WDSBKTP42685	Regular	2019-03-27 08:29:45	KA21A3643	Mugab Bangalore Ru Karnat
6879	JTECH	WDSBKTP42858	Regular	2019-03-28 17:55:17	KA51D1317	Mugab Bangalore Ru Karnat

5 rows × 32 columns

dataframe.info()

This will tell us the total number of non null observations present including the total number of entries. Once number of entries isn't equal to number of non null observations, we can begin to suspect missing values.

Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data-centric python packages. Pandas is one of those packages and makes importing and analyzing data much easier.

Pandas dataframe.info() function is used to get a concise summary of the dataframe. It comes really handy when doing exploratory analysis of the data. To get a quick overview of the dataset we use the dataframe.info() function.

In [8]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6880 entries, 0 to 6879
Data columns (total 32 columns):
 #   Column                          Non-Null Count  Dtype         
---  ------                          --------------  -----         
 0   GpsProvider                     5927 non-null   object        
 1   BookingID                       6880 non-null   object        
 2   Market/Regular                  6880 non-null   object        
 3   BookingID_Date                  6880 non-null   datetime64[ns]
 4   vehicle_no                      6880 non-null   object        
 5   Origin_Location                 6880 non-null   object        
 6   Destination_Location            6880 non-null   object        
 7   Org_lat_lon                     6880 non-null   object        
 8   Des_lat_lon                     6880 non-null   object        
 9   Data_Ping_time                  5927 non-null   datetime64[ns]
 10  Planned_ETA                     6880 non-null   datetime64[ns]
 11  Current_Location                5916 non-null   object        
 12  DestinationLocation             6880 non-null   object        
 13  actual_eta                      6843 non-null   datetime64[ns]
 14  Curr_lat                        5927 non-null   float64       
 15  Curr_lon                        5927 non-null   float64       
 16  ontime                          2548 non-null   object        
 17  delay                           4342 non-null   object        
 18  OriginLocation_Code             6877 non-null   object        
 19  DestinationLocation_Code        6853 non-null   object        
 20  trip_start_date                 6880 non-null   datetime64[ns]
 21  trip_end_date                   6686 non-null   datetime64[ns]
 22  TRANSPORTATION_DISTANCE_IN_KM   6168 non-null   float64       
 23  vehicleType                     6052 non-null   object        
 24  Minimum_kms_to_be_covered_in_a_day  2820 non-null   float64       
 25  Driver_Name                     3451 non-null   object        
 26  Driver_MobileNo                 2691 non-null   float64       
 27  customerID                      6880 non-null   object        
 28  customerNameCode                6880 non-null   object        
 29  supplierID                      6880 non-null   object        
 30  supplierNameCode                6880 non-null   object        
 31  Material Shipped                6880 non-null   object        
dtypes: datetime64[ns](6), float64(5), object(21)
memory usage: 1.7+ MB
```

- 14 Curr_lat 5927 non-null float64
- 15 Curr_lon 5927 non-null float64
- 22 TRANSPORTATION_DISTANCE_IN_KM 6168 non-null float64
- 24 Minimum_kms_to_be_covered_in_a_day 2820 non-null float64
- 26 Driver_MobileNo 2691 non-null float64
- We need to ENCODING for categorical('object') variable and also We need to some manupulating for NUMERICAL variable.

In [9]:

```
#making a copy of data before preprocessing
data_raw=data.copy()
```

What is pandas profile report?

Generates profile reports from a pandas DataFrame . The pandas df.describe () function is great but a little basic for serious exploratory data analysis. pandas_profiling extends the pandas DataFrame with df.profile_report () for quick data analysis.

The pandas_profiling library in Python include a method named as ProfileReport () which generate a basic report on the input DataFrame. A sample of DataFrame. Number of bins in histogram. The default is 10. Whether or not to check correlation. It's True by default. Threshold to determine if the variable pair is correlated. The default is 0.9.

pandas_profiling.ProfileReport(data)

In [9]:
```
#pandas_profiling.ProfileReport(data)
```

The statistical summary of the dataset

This is SUMMERIZING the FACT. This part does not have ESTIMATION.

In [10]:
```
# Getting the summary of Data
data.describe()# for numeric columns
pd.options.display.float_format = "{:.2f}".format
data.describe().transpose()
```

Out[10]:

	count	mean	std	min	
Curr_lat	5927.00	18.68	6.08	8.17	
Curr_lon	5927.00	78.76	4.22	69.66	
TRANSPORTATION_DISTANCE_IN_KM	6168.00	553.86	758.98	0.00	
Minimum_kms_to_be_covered_in_a_day	2820.00	250.24	24.32	0.00	
Driver_MobileNo	2691.00	8598981266.45	1131668748.29	6000546262.00	7651

We need to DESCRIPTIVE DATA, this is some of fact and does NOT MAKE SENSE.

In [11]:

```
#finding count (number of non_missing values),unique values(or levels), top(
#Method 1
data.astype('object').describe().transpose()
```

Out[11]:

	count	unique	top
GpsProvider	5927	29	CONSENT TRACK
BookingID	6880	6875	MVCV0000798/082021
Market/Regular	6880	2	Regular
BookingID_Date	6880	6005	2020-08-12 13:02:21
vehicle_no	6880	2325	TS15UC9341
Origin_Location	6880	180	Mugabala, Bangalore Rural, Karnataka
Destination_Location	6880	520	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...
Org_lat_lon	6880	173	16.560192249175344,80.792293091599547
Des_lat_lon	6880	522	12.8390,79.9540
Data_Ping_time	5927	3756	2019-06-15 11:40:12
Planned_ETA	6880	6294	2020-08-13 09:52:17
Current_Location	5916	2567	Perumalpattu - Kottamedu Rd, Oragadam Industri...
DestinationLocation	6880	520	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...
actual_eta	6843	6729	2020-06-19 18:52:00
Curr_lat	5927.00	4139.00	17.46
Curr_lon	5927.00	4109.00	78.20
ontime	2548	1	G
delay	4342	1	R
OriginLocation_Code	6877	178	V0048673
DestinationLocation_Code	6853	478	CHEMMNFILCCA1
trip_start_date	6880	6107	2020-08-12 13:02:21
trip_end_date	6686	4722	2019-11-11 08:08:00
TRANSPORTATION_DISTANCE_IN_KM	6168.00	564.00	25.00
vehicleType	6052	44	40 FT 3XL Trailer 35MT
Minimum_kms_to_be_covered_in_a_day	2820.00	3.00	250.00
Driver_Name	3451	1355	MANU
Driver_MobileNo	2691.00	1273.00	9952349318.00
customerID	6880	39	LTLEXMUM40
customerNameCode	6880	39	Larsen & toubro limited
supplierID	6880	321	999
supplierNameCode	6880	309	Unknown

	count	unique	top
Material Shipped	6880	1407	AUTO PARTS

#Method 2 #finding just unique values data.apply(lambda x: len(x.unique()))

Data Preprocessing and Data Manipulation

Real-world data is messy. That's why libraries like pandas are so valuable. Using pandas you can take the pain out of data manipulation by extracting, filtering, and transforming data in DataFrames, clearing a path for quick and reliable data analysis.

Manipulation in Python

Part 1

1-1 Working with Rows: Dropping rows
1-2 Working with Columns: Dropping columns
Keeping columns
Adding new columns to a DataFrame
1-3 User defined functions: Creating a new column using functions 1-4 Cleaning dataset: Creating three datasets
Keeping columns
Renaming columns
1-5 Joining/Combining DataFrames and Groupby: Merge on columns
Grouping — Applying an aggregating function
1-6 Graphs and Statistics

Part 2

2-1 Working with Rows Sorting DataFrame rows values
Select a slice of rows by integer position
2-2 Working with Columns Find index label for min/max values in column
Maths on the whole DataFrame
Apply numpy mathematical functions to columns
2-3 Working with cells Selecting a cell by row and column labels
User defined functions
Creating a new column using functions
Find index label for min/max values in column
2-4 Working with missing values and strings Drop all rows with NaN
Drop all columns with NaN
Drop all rows where NaN appear more than twice
Drop all rows where NaN appear in a special column
Recoding all missing data
df.fillna(0, inplace=True)
Recoding missing data in a special column
s = df['AVERAG'].fillna(0)
Working with strings

s = df['col'].str.lower()
s = df['col'].str.upper()
s = df['col'].str.len()
s = df['col'].str.replace('old', 'new')
2-5 Pivot Tables

* DATA PREPROCESSING - HANDLING DUPLICATE DATA

In real world you ara not allowd to remove any obsevation that belongs to test (future) data set, because we have to predict for each observation of tast data set. that's why I will just remove duplicate data from train data set. BUT in this project is removed the observation that has more than 80 percent missing value, and this situation I do not have idea to use that. If we have domain knowledge about project in the feild of work, we can do well with how to deal with data.

In [4]:
```
#drop duplicate
data=data.drop_duplicates()
```

* DATA PREPROCESSING - HANDLING MISSING VALUE

There are several options for handling missing values. However, the choice of what should be done is largely dependent on the nature of our data and the missing values. Below is a summary highlight of several options we have for handling missing values. DROP MISSING VALUES FILL MISSING VALUES WITH TEST STATISTIC PREDICT MISSING VALUE WITH A MACHINE LEARNING ALGORITHM Below is a few list of commands to detect missing values with EDA.

In [5]:

```python
#Python, pandas
#Count missing values for each column of the dataframe df
#By default (axis = 0)

data.isna().sum()
```

Out[5]:

```
GpsProvider                          953
BookingID                              0
Market/Regular                         0
BookingID_Date                         0
vehicle_no                             0
Origin_Location                        0
Destination_Location                   0
Org_lat_lon                            0
Des_lat_lon                            0
Data_Ping_time                       953
Planned_ETA                            0
Current_Location                     964
DestinationLocation                    0
actual_eta                            37
Curr_lat                             953
Curr_lon                             953
ontime                              4332
delay                               2538
OriginLocation_Code                    3
DestinationLocation_Code              27
trip_start_date                        0
trip_end_date                        194
TRANSPORTATION_DISTANCE_IN_KM        712
vehicleType                          828
Minimum_kms_to_be_covered_in_a_day  4060
Driver_Name                         3429
Driver_MobileNo                     4189
customerID                             0
customerNameCode                       0
supplierID                             0
supplierNameCode                       0
Material Shipped                       0
dtype: int64
```

Count total missing values in a dataframe data.isnull().sum().sum()

#Gives a integer value

#Python, pandas #Count missing values for each column of the dataframe df data.isnull().sum(axis = 0)

#Python, pandas #Count missing values for each row of the dataframe df data.isnull().sum(axis = 1)

In [6]:

```python
missingrows = data.isna().sum()
```

In [7]:

```
print(data.columns)
print(missingrows.shape)
print(data.shape)
```

```
Index(['GpsProvider', 'BookingID', 'Market/Regular ', 'BookingID_Date',
       'vehicle_no', 'Origin_Location', 'Destination_Location', 'Org_lat_lon',
       'Des_lat_lon', 'Data_Ping_time', 'Planned_ETA', 'Current_Location',
       'DestinationLocation', 'actual_eta', 'Curr_lat', 'Curr_lon', 'ontime',
       'delay', 'OriginLocation_Code', 'DestinationLocation_Code',
       'trip_start_date', 'trip_end_date', 'TRANSPORTATION_DISTANCE_IN_KM',
       'vehicleType', 'Minimum_kms_to_be_covered_in_a_day', 'Driver_Name',
       'Driver_MobileNo', 'customerID', 'customerNameCode', 'supplierID',
       'supplierNameCode', 'Material Shipped'],
      dtype='object')
(32,)
(6880, 32)
```

In [8]:

```python
#Percentange of missing values
for column in data.columns:
    print('Percentage of missing values in {} is {}'.format(column,missingro
```

```
Percentage of missing values in GpsProvider is 0.1385174418604651
Percentage of missing values in BookingID is 0.0
Percentage of missing values in Market/Regular  is 0.0
Percentage of missing values in BookingID_Date is 0.0
Percentage of missing values in vehicle_no is 0.0
Percentage of missing values in Origin_Location is 0.0
Percentage of missing values in Destination_Location is 0.0
Percentage of missing values in Org_lat_lon is 0.0
Percentage of missing values in Des_lat_lon is 0.0
Percentage of missing values in Data_Ping_time is 0.1385174418604651
Percentage of missing values in Planned_ETA is 0.0
Percentage of missing values in Current_Location is 0.14011627906976745
Percentage of missing values in DestinationLocation is 0.0
Percentage of missing values in actual_eta is 0.005377906976744186
Percentage of missing values in Curr_lat is 0.1385174418604651
Percentage of missing values in Curr_lon is 0.1385174418604651
Percentage of missing values in ontime is 0.6296511627906977
Percentage of missing values in delay is 0.3688953488372093
Percentage of missing values in OriginLocation_Code is 0.00043604651162790697
Percentage of missing values in DestinationLocation_Code is 0.00392441860465116
3
Percentage of missing values in trip_start_date is 0.0
Percentage of missing values in trip_end_date is 0.028197674418604651
Percentage of missing values in TRANSPORTATION_DISTANCE_IN_KM is 0.103488372093
02325
Percentage of missing values in vehicleType is 0.12034883720930233
Percentage of missing values in Minimum_kms_to_be_covered_in_a_day is 0.5901162
790697675
Percentage of missing values in Driver_Name is 0.4984011627906977
Percentage of missing values in Driver_MobileNo is 0.6088662790697674
Percentage of missing values in customerID is 0.0
Percentage of missing values in customerNameCode is 0.0
Percentage of missing values in supplierID is 0.0
Percentage of missing values in supplierNameCode is 0.0
Percentage of missing values in Material Shipped is 0.0
```

* DATA PREPROCESSING - DROP MISSING VALUE

DROP the data NOT CORUPPTED but, FILLING MISSING data is CORRUPTED the data.

In [9]:

```python
#drop columns that have 80% or more missing values
data = data.dropna(axis=1, thresh=1376)
```

Cleaning Data : dropna() thresh option

Keep only columns where 80% or more valid data is available

thresh= 6880*0.20= 1376

* DATA PREPROCESSING - FILLING MISSING VALUE

pandas.Series.rolling

Series.rolling(window, min_periods=None, center=False, win_type=None, on=None, axis=0, closed=None)[source] Provide rolling window calculations.

Parameters:

windowint, offset, or BaseIndexer subclass: Size of the moving window. This is the number of observations used for calculating the statistic. Each window will be a fixed size. If its an offset then this will be the time period of each window. Each window will be a variable sized based on the observations included in the time-period. This is only valid for datetimelike indexes. If a BaseIndexer subclass is passed, calculates the window boundaries based on the defined get_window_bounds method. Additional rolling keyword arguments, namely min_periods, center, and closed will be passed to get_window_bounds.

min_periodsint, default None:

Minimum number of observations in window required to have a value (otherwise result is NA). For a window that is specified by an offset, min_periods will default to 1. Otherwise, min_periods will default to the size of the window.

centerbool, default False:

Set the labels at the center of the window.

***A rolling mean is simply the mean of a certain number of previous periods in a time series.

To calculate the rolling mean for one or more columns in a pandas DataFrame, we can use the following syntax:

df['column_name'].rolling(rolling_window).mean()

In [10]:

```
#Method1
data.fillna({
            #name unkown for null values in driver name
            'Driver_Name' : data.Driver_Name.fillna('Unknown'),

            #impute transportation distence with mean value
            'TRANSPORTATION_DISTANCE_IN_KM': data.TRANSPORTATION_DISTANCE_IN

            #name unkown for null values in vehicle type
            'vehicleType':data.vehicleType.fillna('Unknown'),

            #fill pervious date for actual.eta
            'actual_eta':data.actual_eta.fillna(method='ffill'),
            },
        inplace = True)


```

#Method2

#let's check the percentage of null values in each feature for col in data.columns: if data[col].isna().sum()>0: print(col, data[col].isna().mean().round(4)*100)

#Method2

#name unkown for null values in driver name data['Driver_Name']=data['Driver_Name'].fillna('Unknown')

#impute transportation distence with mean value data['TRANSPORTATION_DISTANCE_IN_KM']= data["TRANSPORTATION_DISTANCE_IN_KM"].rolling(min_periods=1, center=True, window=12).mean()

#name unkown for null values in vehicle type data['vehicleType']=data['vehicleType'].fillna('Unknown')

#fill pervious date for actual.eta data['actual_eta']=data['actual_eta'].fillna(method='ffill')

* DATA PREPROCESSING - CREATE A COLUMN

In [11]:

```
#Method1
#create as a single column 'ontime/delay' from 'ontime' and 'delay' columns
data['ontime/delay']=data.ontime.replace({np.NaN, 'G'}, {0, 1})
```

In [20]:
```
data['ontime/delay']
```

Out[20]:
```
0       0
1       1
2       1
3       1
4       1
       ..
6875    1
6876    1
6877    1
6878    0
6879    1
Name: ontime/delay, Length: 6880, dtype: int64
```

#Method2 #create as a single column 'ontime/delay' from 'ontime' and 'delay' columns

data['ontime/delay']=data['ontime'].replace(np.NaN, 'G')

data['ontime/delay']=data['delay'].replace(np.NaN, 'G')

data['ontime/delay'] =data['ontime/delay'].map({ 'G': 1, 'R': 0 })

data['ontime/delay']

* DATA PREPROCESSING - A SIMPLE APPLICTION OF MECHINE LEARNING

vehicle_no	Origin_Location	Destination_Location
KA590408	TVSLSL-PUZHAL-HUB,CHENNAI,TAMIL NADU	ASHOK LEYLAND PLANT 1- HOSUR,HOSUR,KARNATAKA
TN30BC5917	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
TN22AR2748	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY
TN28AQ0781	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
TN68F1722	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY
TN88A4980	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
TN88C8204	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
TN88D4133	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
TN23AM4662	ASHOK LEYLAND ENNORE,CHENNAI,TAMIL NADU	ASHOK LEYLAND PLANT 2-HOSUR,HOSUR,KARNATAKA
TN30BC5982	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
TN88D3900	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,TAMIL NADU
KA51C6972	ASHOK LEYLAND PLANT 2-HOSUR,HOSUR,KARNATAKA	TVSLSL-PUZHAL-HUB,CHENNAI,TAMIL NADU

- Using the str() constructor to make a string
- Since strings are arrays, we can loop through the characters in a string, with a for loop
- The split() method splits the string into substrings if it finds instances of the separator and returns a list

- IN MECHINE LEARNING course we deep on that

In [12]:
```python
# converting dtypes using astype
# getting the first 2 letters of a string in 'vehicle_states' for mention th
data['vehicle_states'] = data.vehicle_no.astype(str).str[:2]
data['Origin_states'] = data['Origin_Location'].str.split(',').apply(lambda
data['Dest_states'] = data['Destination_Location'].str.split(',').apply(lamb
```

data['vehicle_states']=data['vehicle_states'].replace(('tn', 'hr'), ('TN', 'HR'))

In [13]:
```python
pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\sarak\anaconda3\lib\site-packa
ges (2.1.0)
Requirement already satisfied: geographiclib<2,>=1.49 in c:\users\sarak\anacond
a3\lib\site-packages (from geopy) (1.50)
Note: you may need to restart the kernel to use updated packages.
```

```
pandas.DataFrame.itertuples

DataFrame.itertuples(index=True, name='Pandas')[source]
Iterate over DataFrame rows as namedtuples.

Parameters
indexbool, default True
If True, return the index as the first element of the tuple.
```

By setting the *index* parameter to False we can remove the index as the first element of the tuple:

```
>>> for row in df.itertuples(index=False):
...     print(row)
...
Pandas(num_legs=4, num_wings=0)
Pandas(num_legs=2, num_wings=2)
```

In [14]:
```python

from geopy import distance

#find the distance between origin and destination
distances_km = []
for row in data.itertuples(index=False):
   distances_km.append(
       distance.distance(row.Org_lat_lon, row.Des_lat_lon).km
   )

data['Org_Dest_distance'] = distances_km
#df_dist.head()

#data=pd.concat([data, df_dist])
```

In [15]:

```
#feature like gps provider, data ping time, current location, curr_lat, curr
#all the above mentioned features are dependent on each other and it's not f

data.dropna(subset=['Current_Location'], inplace=True)
```

In [16]:

```
#checking whether having driver's mobile number making any impact on ontime
data['Driver_MobileNo'].values[data['Driver_MobileNo'].values>0]=1
data['Driver_MobileNo'].fillna(0, inplace=True)
```

DATA PREPROCESSING - Feature Encoding

- ENCODING means CHANGE STRING values to NUMBERS.

In [17]:

```
#filtering data
df_cln=data[['Market/Regular ',
        'vehicle_no',
        'Current_Location',
        'TRANSPORTATION_DISTANCE_IN_KM',
        'vehicleType', 'Driver_Name',
        'Driver_MobileNo', 'customerID', 'supplierID',
        'Material Shipped', 'ontime/delay',
        'vehicle_states', 'Origin_states', 'Dest_states', 'Org_Dest_distance'
```

- Most of the data, are NOMINAL and CATEGORICAL.

In [18]:

```
#making a copy of filtered data
df_copy=df_cln.copy()
```

* DATA PREPROCESSING - Mapping Categorical Data in pandas

- In python, unlike R, there is no option to represent categorical data as factors. Factors in R are stored as vectors of integer values and can be labelled.
- If we have our data in Series or Data Frames, we can convert these categories to numbers using pandas Series' astype method and specify 'categorical'.

***Nominal Categories:

Nominal categories are unordered e.g. colours, sex, nationality. In the example below we categorise the Series vertebrates of the df dataframe into their individual categories. By default the categories are ordered alphabetically, which is why in the example below Amphibian is represented by a zero.

- import pandas as pd
- df = pd.DataFrame({'vertebrates': ['Bird', 'Bird', 'Mammal', 'Fish', 'Amphibian', 'Reptile', 'Mammal']})
- df.vertebrates.astype("category").cat.code

df.vertebrates.astype("category").cat.codes

- 0 1
- 1 1
- 2 3
- 3 2
- 4 0
- 5 4
- 6 3 dtype: int8

***If we wanted to separate the distinct variables out into booleans as we would like for data science models such as, for example, linear regression, we can use pd.get_dummies.

---

pd.get_dummies(df, columns=['vertebrates'])

vertebrates_Amphibian vertebrates_Bird vertebrates_Fish vertebrates_Mammal vertebrates_Reptile

- 0 0 1 0 0 0
- 1 0 1 0 0 0
- 2 0 0 0 1 0
- 3 0 0 1 0 0
- 4 1 0 0 0 0
- 5 0 0 0 0 1
- 6 0 0 0 1 0

***Ordinal Categories:

Ordinal categories are ordered, e.g. school grades, price ranges, salary bands. For ordinal categorical data, you pass the parameter ordered = True to the astype method. ordered_satisfaction = ['Very Unhappy', 'Unhappy', 'Neutral', 'Happy', 'Very Happy'] df = pd.DataFrame({'satisfaction':['Mad', 'Happy', 'Unhappy', 'Neutral']}) We can have the output categories as text, with NaN for any missing categories: df.satisfaction.astype("category", ordered=True, categories=ordered_satisfaction)

- 0 NaN
- 1 Happy
- 2 Unhappy
- 3 Neutral

Name: satisfaction, dtype: category

Categories (5, object): [Very Unhappy < Unhappy < Neutral < Happy < Very Happy] Or the output categories as numbers that map to the ordered categories. The number -1 is given to any missing category.

df.satisfaction.astype("category", ordered=True, categories=ordered_satisfaction).cat.codes

- 0 -1
- 1 3
- 2 1
- 3 2

dtype: int8

In [19]:
```
df_cln['supplierID']=df_cln.supplierID.astype("category").cat.codes
df_cln['Dest_states']=df_cln['Dest_states'].astype("category").cat.codes
df_cln['Origin_states']=df_cln['Origin_states'].astype("category").cat.codes
df_cln['vehicleType']=df_cln.vehicleType.astype("category").cat.codes
df_cln['vehicle_states']=df_cln.vehicle_states.astype("category").cat.codes
df_cln['vehicle_no']=df_cln.vehicle_no.astype("category").cat.codes
df_cln['Current_Location']=df_cln.Current_Location.astype("category").cat.co
df_cln['Material Shipped']=df_cln['Material Shipped'].astype("category").cat
df_cln['Market/Regular ']=df_cln['Market/Regular '].astype("category").cat.c
df_cln['Driver_Name']=df_cln['Driver_Name'].astype("category").cat.codes
df_cln['customerID']=df_cln.customerID.astype("category").cat.codes
```

Exploratory Analysis - Corrolation

In [20]:
```
corr_matrix=df_cln.corr()
corr_matrix["ontime/delay"].sort_values(ascending=False)
```

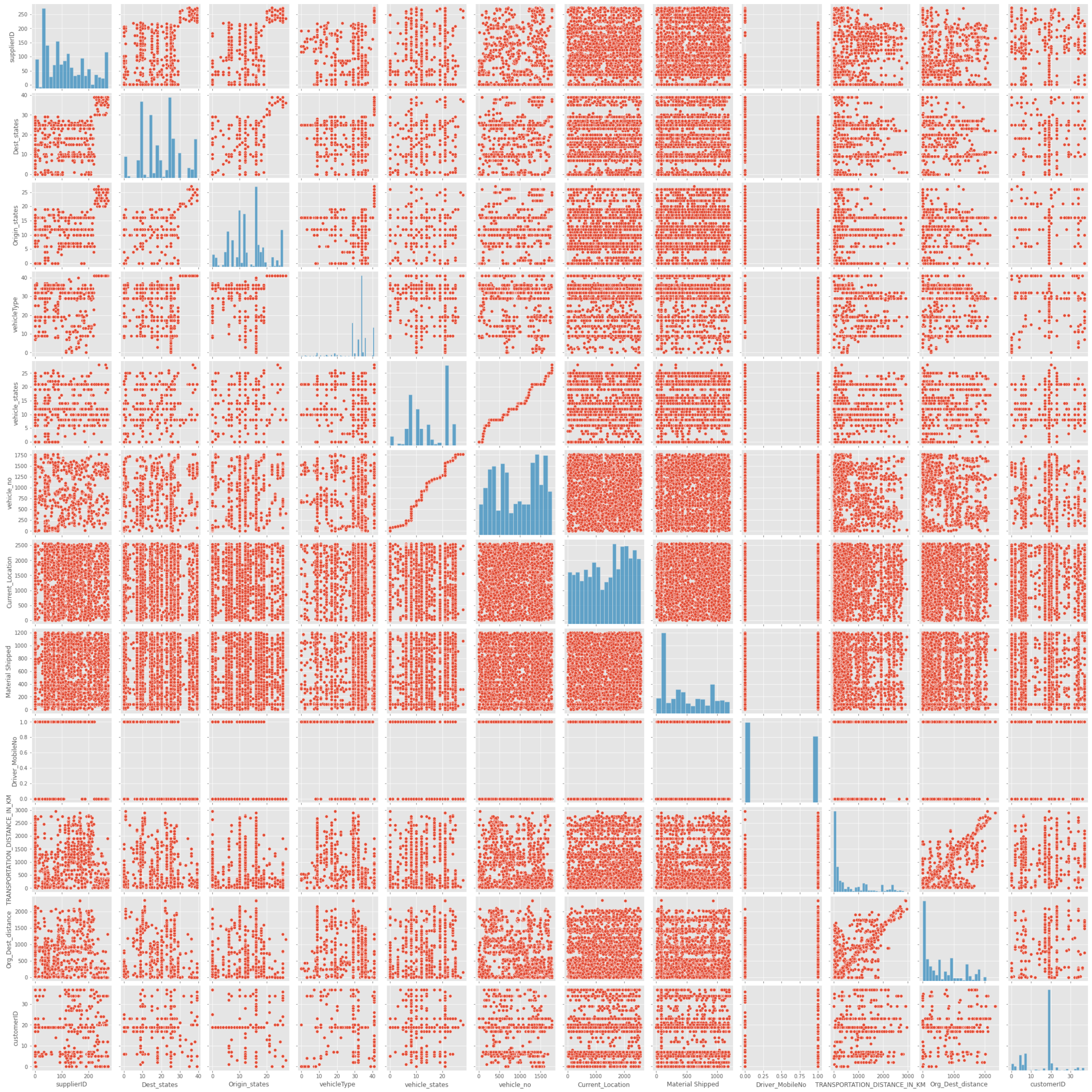
Out[20]:
```
ontime/delay                  1.000000
supplierID                    0.501161
Dest_states                   0.479652
Origin_states                 0.443254
vehicleType                   0.210687
vehicle_states                0.205373
vehicle_no                    0.201921
Current_Location              0.075295
Material Shipped              0.013937
Driver_MobileNo              -0.033495
TRANSPORTATION_DISTANCE_IN_KM  -0.077015
Market/Regular               -0.125849
Org_Dest_distance            -0.233716
Driver_Name                  -0.244630
customerID                   -0.258496
Name: ontime/delay, dtype: float64
```

In [21]:
```
df_cln = df_cln.sample(frac=1, random_state=0)
```

In [22]:
```
X=df_cln.drop('ontime/delay', axis=1)
y=df_cln['ontime/delay'].values
```

In [24]:

```python
# Pairplot of all the numeric variables
# Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.pairplot(df_cln, vars=['supplierID','Dest_states','Origin_states','vehic
plt.show()
```

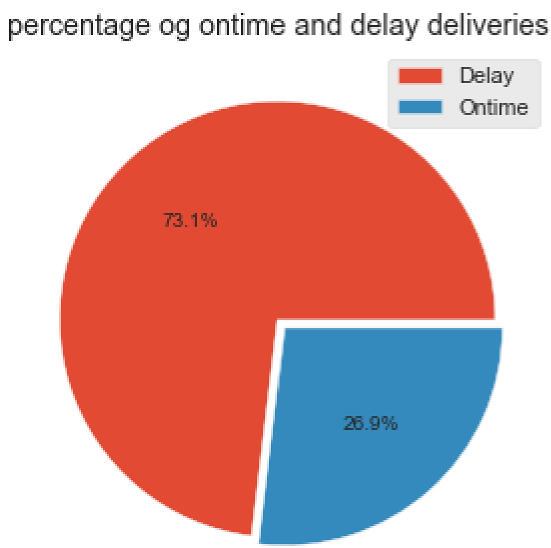


- Correlation and heatmap is used for NUMERICAL VARIABLE , therefore when we want to use that in Logestic Regression, we must drop TARGET in heat map and correlation

Exploratory Analysis - Frequency Distribution

For target variable we can predict visulaization of variable

In [70]:
```
plt.rcParams['figure.figsize']=(5,5)
plt.pie(data['ontime/delay'].value_counts(), explode = (0, 0.05), autopct='%
plt.title('percentage og ontime and delay deliveries')
plt.legend(['Delay', 'Ontime'])
plt.show()
```



#Method2

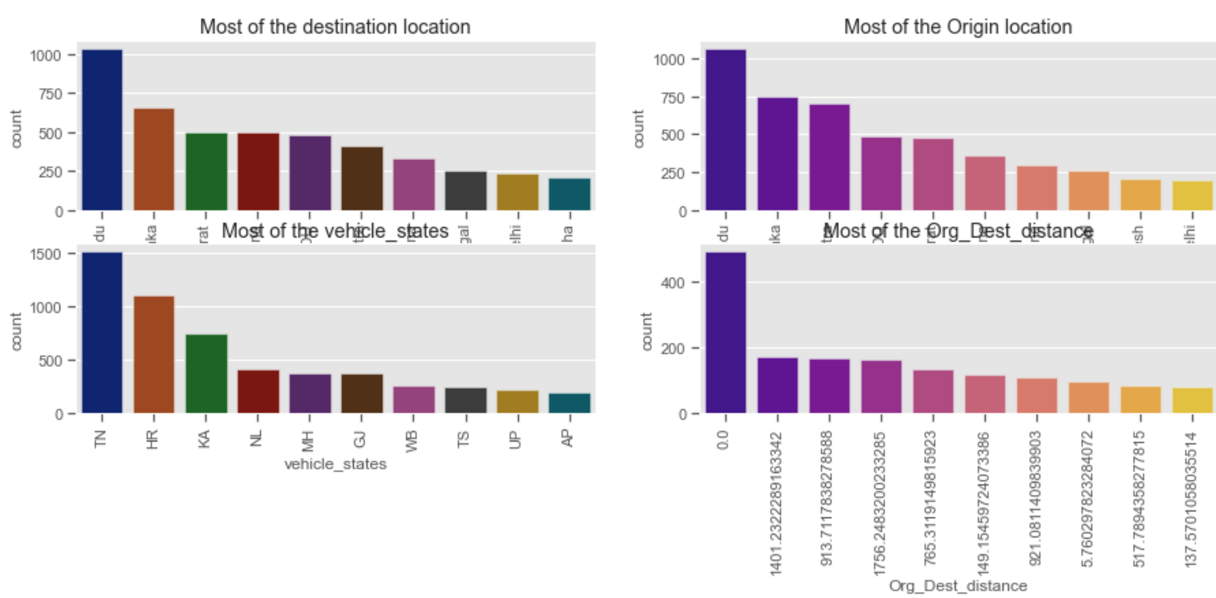
sns.countplot(x=data['ontime/delay'])

sns.despine()

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. For example, you may have a 2-class (binary) classification problem with 100 instances (rows). A total of 80 instances are labeled with Class-1 and the remaining 20 instances are labeled with Class-2.

In [71]:

```python
plt.rcParams['figure.figsize']=15,5

plt.subplot(221)
sns.countplot(data['Dest_states'],
              order=data['Dest_states'].value_counts().head(10).index,
              palette='dark')
plt.xticks(rotation=90)
plt.title('Most of the destination location')

plt.subplot(222)
sns.countplot(data['Origin_states'],
              order=data['Origin_states'].value_counts().head(10).index,
              palette='plasma')
plt.xticks(rotation=90)
plt.title('Most of the Origin location')

plt.subplot(223)
sns.countplot(data['vehicle_states'],
              order=data['vehicle_states'].value_counts().head(10).index,
              palette='dark')
plt.xticks(rotation=90)
plt.title('Most of the vehicle_states')

plt.subplot(224)
sns.countplot(data['Org_Dest_distance'],
              order=data['Org_Dest_distance'].value_counts().head(10).index,
              palette='plasma')
plt.xticks(rotation=90)
plt.title('Most of the Org_Dest_distance')

plt.show()

```

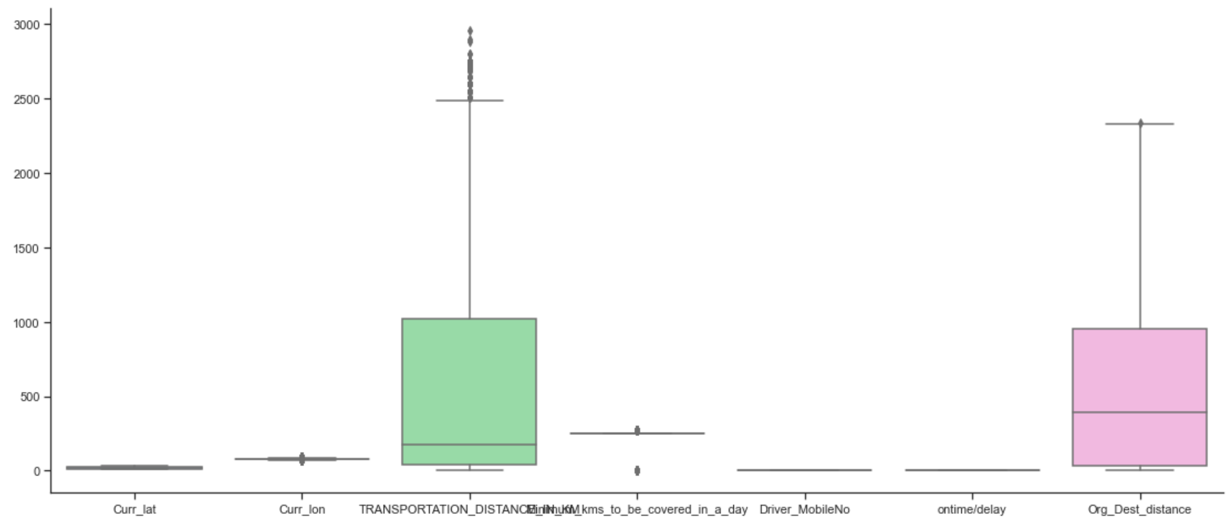


By using visulaization for see the VALUE_COUNT, we can observe the label of graph are very messy, therefore we need to deep in mechine learning.

Exploratory Analysis - FINDING OUTLIER

Using box-plot for display the outlier

In [72]:
```
# Finding outliers by using BOXPLOT
sns.set_theme(style="ticks", palette="pastel")
plt.figure(figsize=(19,8))
sns.boxplot(data=data)
sns.despine()
```



we don't have major outliers in our data

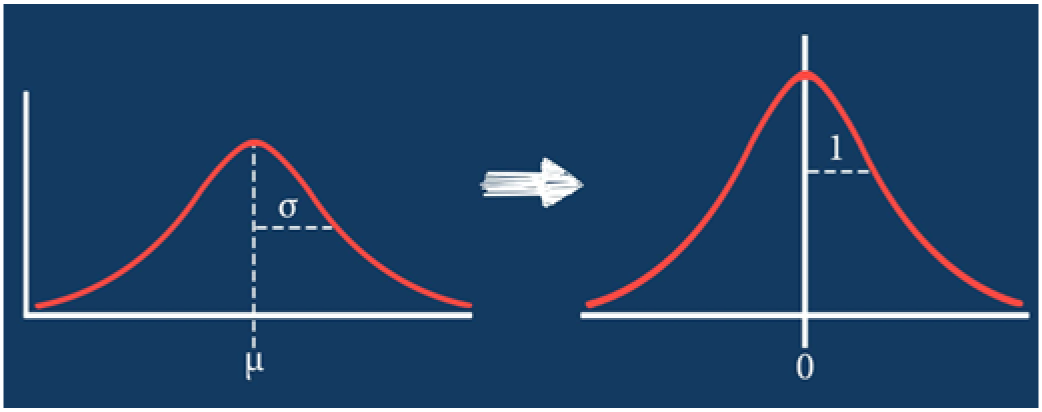
DATA PREPROCESSING - Scaling

The datasets which we use to build a model for a particular problem statement is usually built from various sources. Thus, it can be assumed that the data set contains variables/features of different scales. In order for our machine learning or deep learning model to work well, it is very necessary for the data to have the same scale in terms of the Feature to avoid bias in the outcome.

Thus, Feature Scaling is considered an important step prior to the modeling. Feature Scaling can be broadly classified into the below categories:

Normalization Standardization

The STANDARDIZATION method should be used the data has OUTLIER.



Standardize features by removing the mean and scaling to unit variance.The standard score of a sample x is calculated as:

z = (x - u) / s

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False.

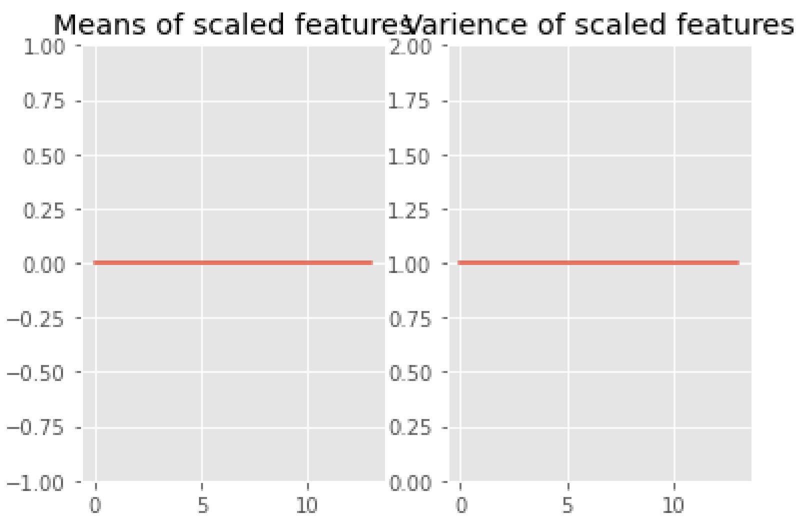
Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform.

In [26]:

```
from sklearn.preprocessing import StandardScaler

#standardizing all the columns
sc=StandardScaler()
scaled=sc.fit_transform(X)

#converted to dataframe to work easily on columns
x_scl=pd.DataFrame(scaled, columns=X.columns)
```

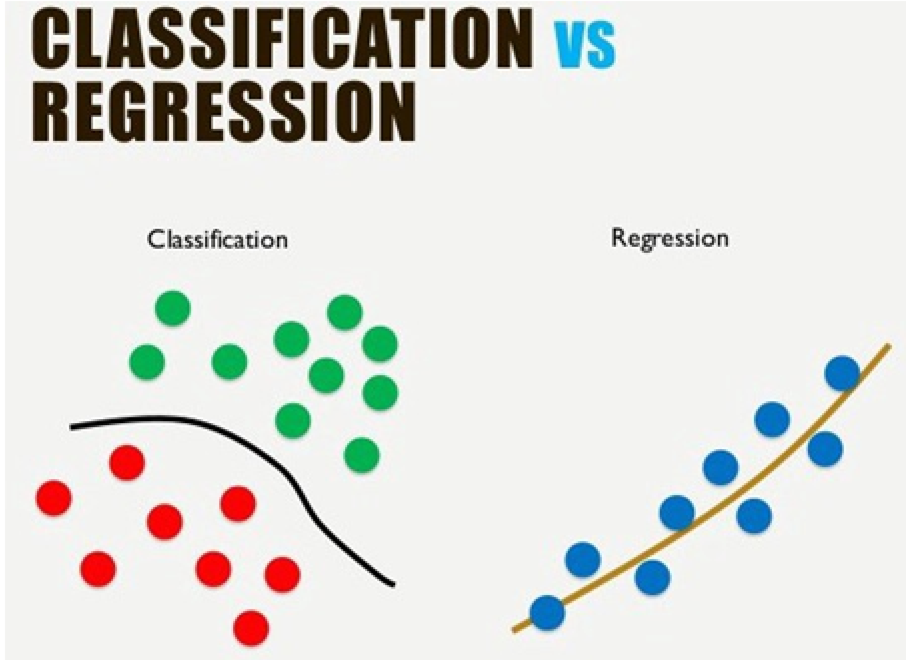
In [27]:

```
#check weathear data is standardized or not
plt.subplot(121)
plt.ylim(-1,1)

means=[]
for i in range(x_scl.shape[1]):
    means.append(np.mean(x_scl.iloc[:,i]))
plt.plot(means, scaley=False)
plt.title('Means of scaled features')

plt.subplot(122)
plt.ylim(0,2)
vars=[]
for i in range(x_scl.shape[1]):
    vars.append(np.var(x_scl.iloc[:,i]))
plt.plot(vars, scaley=False)
plt.title('Varience of scaled features')
plt.show()
```



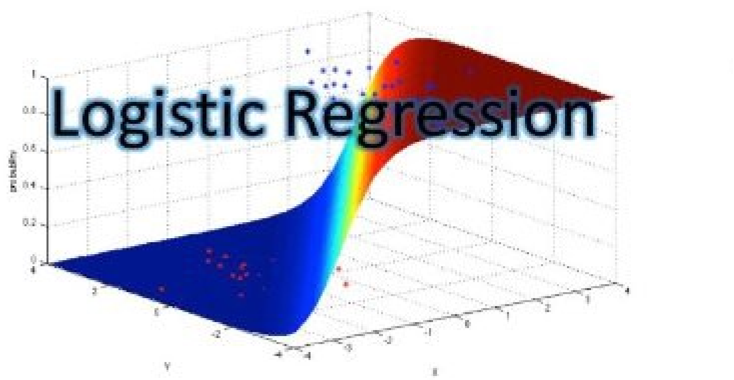
dataset is well standardised

Data Handling - Predictive / Supervised Learning - Logestic Regression



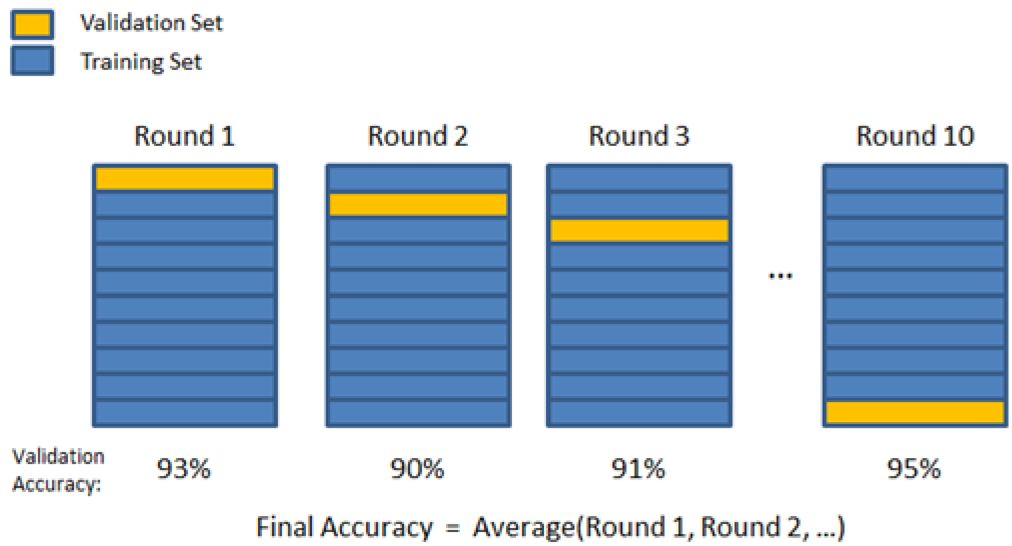
In [28]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_scl, y, test_size=0.25
```

This project has mare than one variable (X), therefore this model IS MULTI- CLASS CLASSIFICATION.



In [29]:
```
#import Logestic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc
```

K Fold Cross Validation



K-Fold CV gives a model with less bias compared to other methods. In K-Fold CV, we have a paprameter ‘ k ’. This parameter decides how many folds the dataset is going to be divided. BUT using K FOLD give us much more time to run. we need to use small size in k fold. In this project k=5 therefore the TEST SIZE is 20%.

In [30]:
```
#k-fold cross-validation
from sklearn.model_selection import cross_val_score
cross_val_score(LogisticRegression(),X,y,cv=5).mean()
```

Out[30]:
```
0.8468599358022434
```

- Before shffling 81.23% with imbalance data.
- After shuffling 84.68% with imbalance data.

In [31]:
```
LR=LogisticRegression()
LR.fit(X_train, y_train)
```

Out[31]:
```
LogisticRegression()
```

In [32]:
```
#To be able to test we need to scale the test data too (X part only)
#using the same scaler that was used to scale the training data
X_test_scaled = sc.transform(X_test)
```

In [33]:
```
#Predict_proba gives the probabilities P(y=Ci|x)
LR.predict_proba(X_test_scaled)
```

Out[33]:
```
array([[0.95694891, 0.04305109],
       [0.95756914, 0.04243086],
       [0.83250121, 0.16749879],
       ...,
       [0.96162562, 0.03837438],
       [0.96206526, 0.03793474],
       [0.95697542, 0.04302458]])
```

In [34]:
```
probabilities_test = LR.predict_proba(X_test_scaled)[:,1]
probabilities_test[10:20] #second column belongs to class 1, ie, p = P(y=1|x
```

Out[34]:
```
array([0.0382833 , 0.16682405, 0.05426274, 0.03828452, 0.1795041 ,
       0.03824654, 0.1743403 , 0.03906766, 0.03907087, 0.15954068])
```

In [35]:
```
#Whereas predict method gives the class prediction as either 0 or 1
y_pred_LR=LR.predict(X_test)
y_pred_LR[10:20]
```

Out[35]:
```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

preprocessing.Binarizer() is a method which belongs to preprocessing module. It plays a key role in the discretization of continuous feature values. The data to binarize, element by element. scipy.sparse matrices should be in CSR or CSC format to avoid an un-necessary copy. Feature values below or equal to this are replaced by 0, above it by 1. Threshold may not be less than 0 for operations on sparse matrices.

We need to match MODEL and BINARIZE

In [36]:
```
#Predictions based on a different threshold value
from sklearn.preprocessing import binarize
y_predict_thresh = binarize(probabilities_test.reshape(-1,1),threshold=0.75)
y_predict_thresh[10:20]
```

Out[36]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

In [38]:
```
#Performance measures for classification
#Accuracy = total no. of correct prediction/total no. of datapoints

LR.score(X_train,y_train)
LR.score(X_test_scaled,y_test)

print(LR.score(X_train,y_train))
print(LR.score(X_test_scaled,y_test))
```

```
0.8519269776876268
0.7261663286004056
```

In [39]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
accuracy_score(y_test,y_pred_LR)
```

Out[39]:
```
0.8343475321162948
```

In [40]:
```
'''
        Predicted
        0   1
True  0 TN  FP
      1 FN  TP

'''
cm1 = confusion_matrix(y_test,y_pred_LR)
cm1
```

Out[40]:
```
array([[1019,   55],
       [ 190,  215]], dtype=int64)
```

In [41]:
```
#Confusion matrix corresponding prob threshold = 0.75
cm2 = confusion_matrix(y_test,y_predict_thresh)
cm2
```

Out[41]:
```
array([[1074,    0],
       [ 405,    0]], dtype=int64)
```

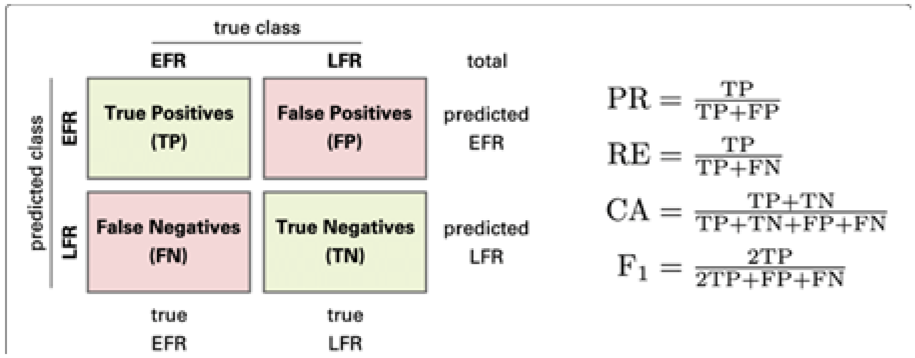
#Fpr = fp/(tn+fp) #tpr = tp/(fn+tp)

fpr1= 26/1069 fpr2 = 7/1033

tpr1 = 160/410 tpr2 = 192/410

In [43]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_LR))
```

```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      1074
           1       0.80      0.53      0.64       405

    accuracy                           0.83      1479
   macro avg       0.82      0.74      0.76      1479
weighted avg       0.83      0.83      0.82      1479
```



- F1 score

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

- Support

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn’t change between models but instead diagnoses the evaluation process.

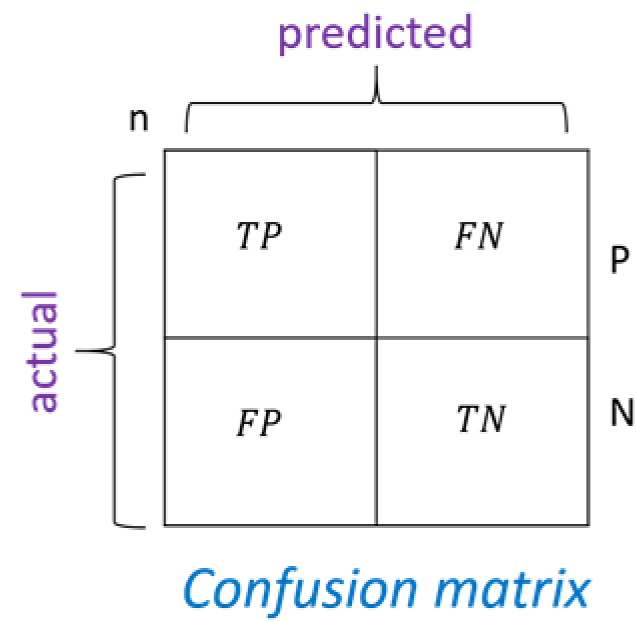
- average=macro says the function to compute f1 for each label, and returns the average without considering the proportion for each label in the dataset.
- average=weighted says the function to compute f1 for each label, and returns the average considering the proportion for each label in the dataset.

MEASURE PERFORMANCE - The confusion matrix

Confusion Matrix mainly used for the classification algorithms which fall under supervised learning. Using the above positive and negative targets information table, we will populate the matrix which gives a much more clear understanding of how the confusion matrix constructed.

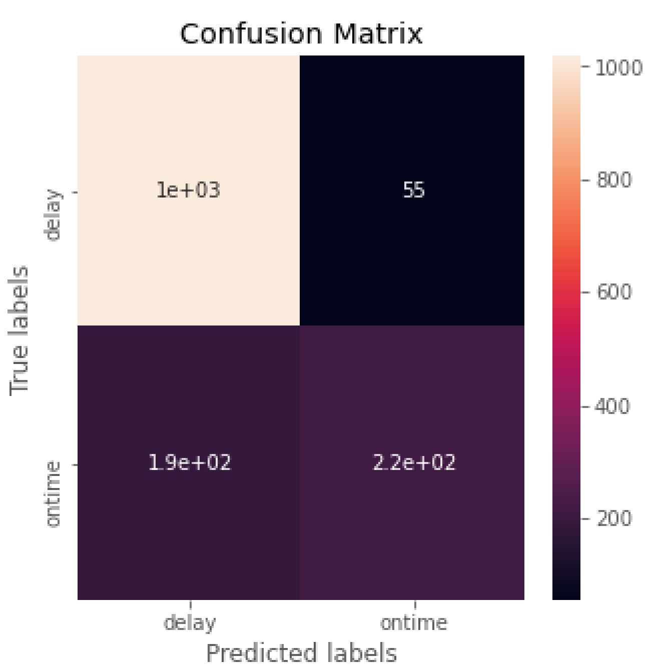
ACCURACY = CORRECT PEREDICTION / TOTAL DATA INSTANCE

Therefore FN and FP are wrong prediction.



Compute confusion matrix to evaluate the accuracy of a classification

In [46]:
```
plt.rcParams['figure.figsize']=5,5
ax= plt.subplot()
cm = confusion_matrix(y_test, y_pred_LR)
sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['delay', 'ontime']); ax.yaxis.set_ticklabels(['dela
```



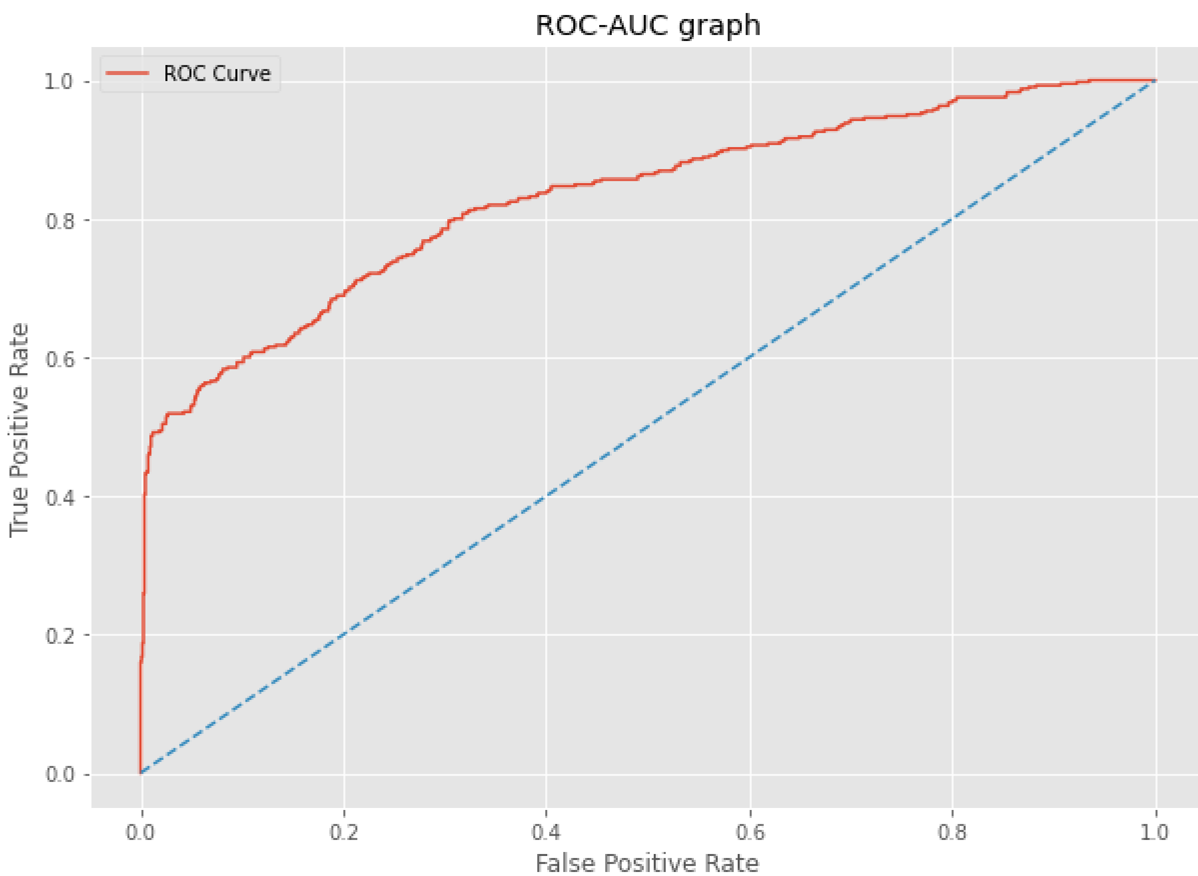
In [47]:

```
#METHOD1:GET ROC GRAPH

#ROC is the curve traced by the co-ordinates (FPR,TPR)
#for different probability threshold values
#AUC is the area under the ROC curve

from sklearn.metrics import roc_auc_score, roc_curve
#y_pred_prob = LR.predict_proba(X_test_scaled)[:,1]

#fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

#METHOD2:GET ROC GRAPH

fpr, tpr, thres = roc_curve(y_test, LR.predict_proba(X_test)[:,1])
roc_auc = auc(fpr, tpr)
```

ROC Curve in Python with Example ROC or Receiver Operating Characteristic curve is used to evaluate logistic regression classification models.

In [48]:

```
plt.rcParams['figure.figsize']=10,7
plt.plot(fpr, tpr, label = 'ROC Curve' %roc_auc)
plt.plot([0, 1], [0, 1], '--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.title('ROC-AUC graph')
plt.show()
```



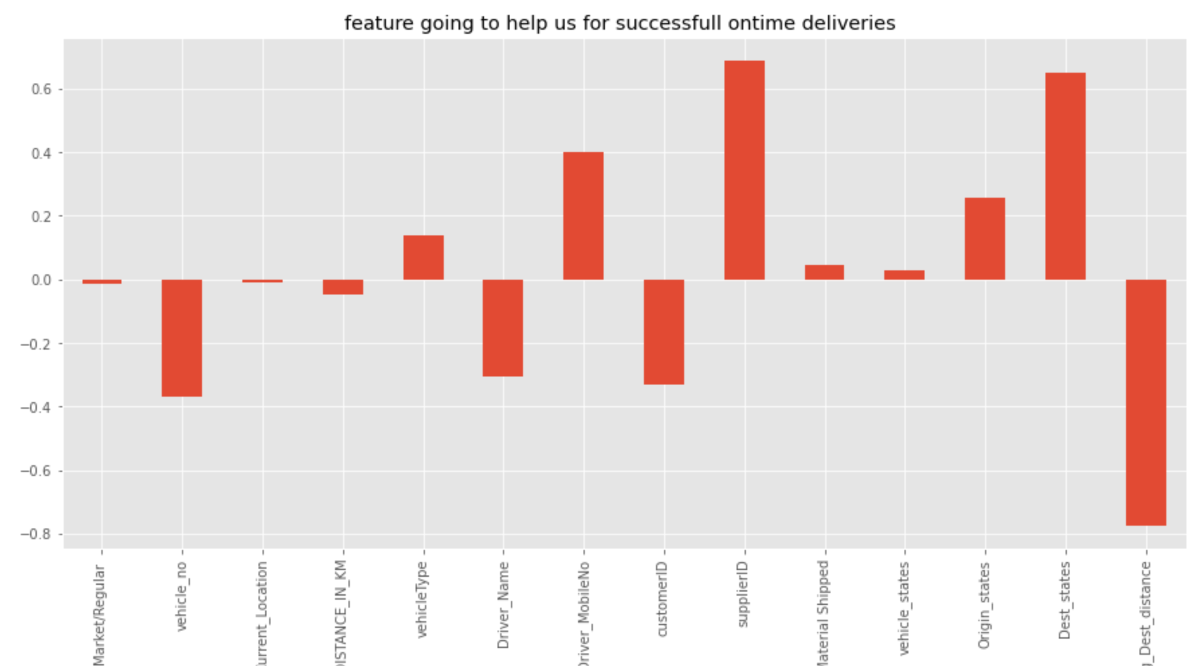
AUC is an excellent performance measure for Logistic Regression Model as it is robust against probability threshold values and truly depicts if the model is good or not for the data at hand. The closer the score to 1, the better. If the score is near 0.5, it means that Logistic Regression is not a good fit for the data. Either we need to get more discriminative features to help identify the target class or look for other model options (may be a complex non-linear model)

```
Selecting most helping Parametes
```

Below we have plotted a bar chart of global feature importance based on weights derived from logistic regression. We can use it to compare it with the bar chart generated for individual data samples.

In [49]:

```python
plt.rcParams['figure.figsize']=15,7
plt.style.use('ggplot')
weights=pd.Series(LR.coef_[0], index=['Market/Regular ', 'vehicle_no', 'Curr
       'TRANSPORTATION_DISTANCE_IN_KM', 'vehicleType', 'Driver_Name',
       'Driver_MobileNo', 'customerID', 'supplierID', 'Material Shipped',
       'vehicle_states', 'Origin_states', 'Dest_states', 'Org_Dest_distance'

params_weight =weights.plot(kind='bar', title='feature going to help us for
plt.style.use('ggplot')
fig=params_weight.get_figure()
plt.show()
```



CONCLUSION

Parameters that impact on ontime delivery:

-Current location

-Transportation distance

-Vehicle state

-Vehicle type

-Driver mobile number

-Supplier

-material shipped

-Destination state

*** We can not VISUALIZATION decision boundary, becuase we have more than three variables and we have HYPER-PLANE. *** In LOGESTIC REGRESSION, if intercept changes the slope will be changed.

- The accuracy of the model is not high, therefore we can apply other models for this project such as, D-TREE, FOREST-TREE and so on.
- If the data is INBALANCED, the ACCURACY is sometimes not the BEST PERFORMANCE.

- To predict the best model when we deal with DATA INBALANCE:

  1) Reduce the inbalance (use oversample, undersample, SMOTE)

  2) Change the probability THRESHOLD IN DECISION RULE.

*** In this project we have a inbalance data, by using class-weight in the logestic regression maybe we handle the inbalance data.