In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
df=pd.read_csv(r"C:\Users\HP\OneDrive\Documents\used_cars_data.csv")
df
```

Out[2]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	
...	...	...	...	...	...	...	...	
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	

7253 rows × 14 columns

In [3]:

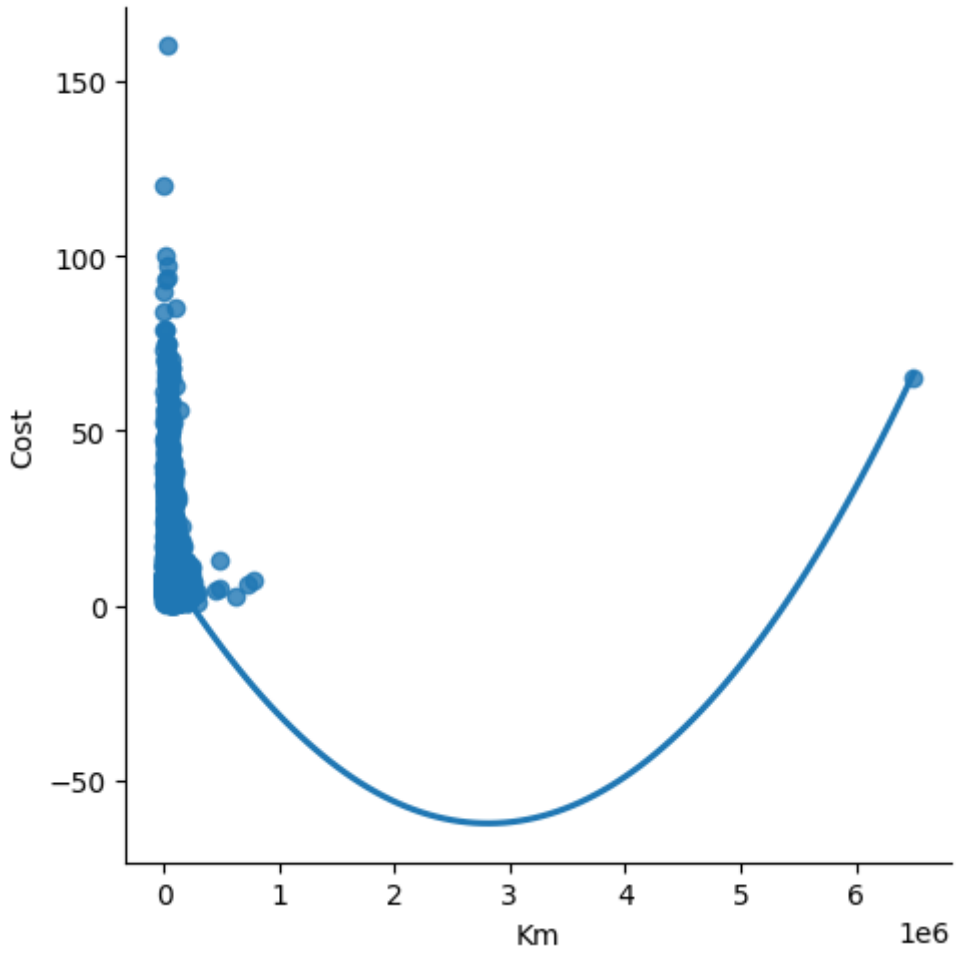
```
df = df[['Kilometers_Driven','Price']]
df.columns=['Km','Cost']
```

In [4]:

```
sns.lmplot(x='Km',y='Cost',data=df,order=2,ci=None)
```

Out[4]:

```
<seaborn.axisgrid.FacetGrid at 0x201d0f19930>
```



In [5]:

```
1 df.describe()
```

Out[5]:

	Km	Cost
count	7.253000e+03	6019.000000
mean	5.869906e+04	9.479468
std	8.442772e+04	11.187917
min	1.710000e+02	0.440000
25%	3.400000e+04	3.500000
50%	5.341600e+04	5.640000
75%	7.300000e+04	9.950000
max	6.500000e+06	160.000000

In [6]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Km      7253 non-null   int64  
 1   Cost    6019 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 113.5 KB
```

In [7]:

```
1 df.fillna(method='ffill',inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_8628\4116506308.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [8]:

```
1 df.dropna(inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_8628\1379821321.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

In [9]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Km      7253 non-null   int64
 1   Cost    7253 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 113.5 KB
```

In [10]:

```
1 df.isnull().sum()
```

Out[10]:

```
Km      0
Cost    0
dtype: int64
```

In [11]:

```
1 df.head(10)
```

Out[11]:

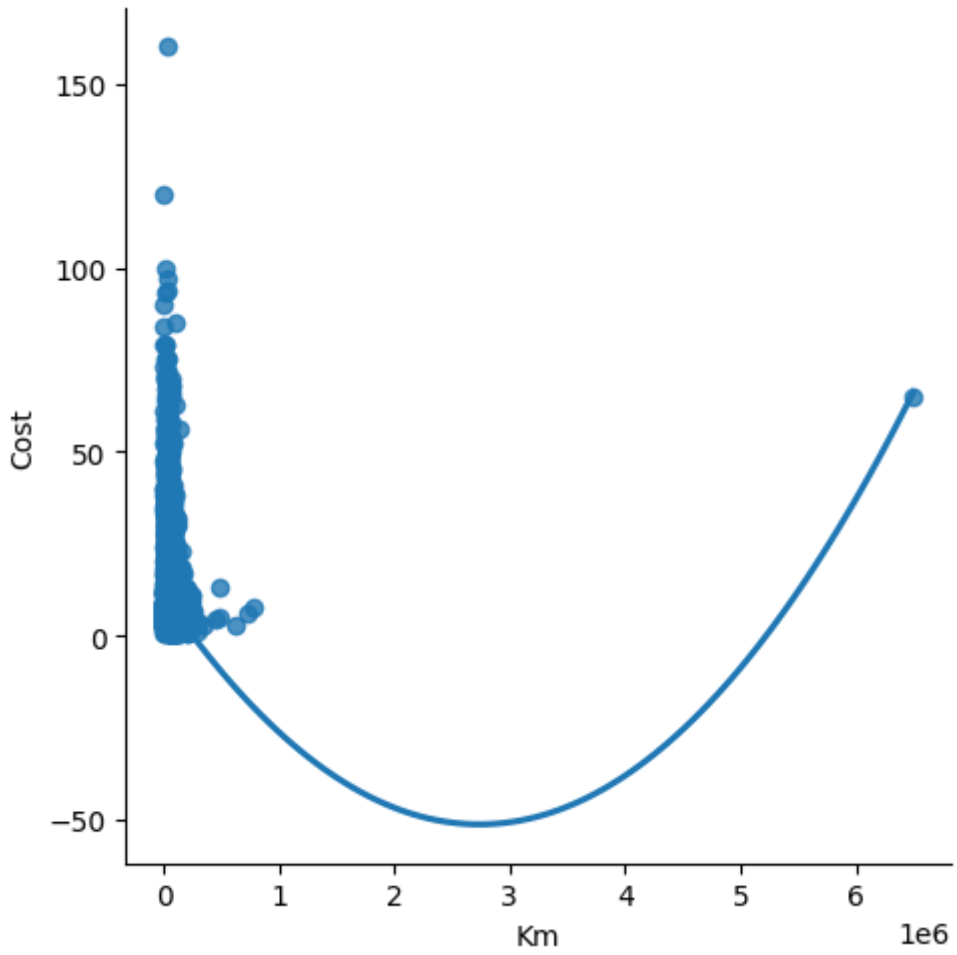
	Km	Cost
0	72000	1.75
1	41000	12.50
2	46000	4.50
3	87000	6.00
4	40670	17.74
5	75000	2.35
6	86999	3.50
7	36000	17.50
8	64430	5.20
9	65932	1.95

In [12]:

```
1 sns.lmplot(x='Km',y='Cost',data=df,order=2,ci=None)
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x201b6d18370>
```



In [13]:

```
x=np.array(df['Km']).reshape(-1,1)
y=np.array(df['Cost']).reshape(-1,1)
```

In [14]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25)
regr = LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

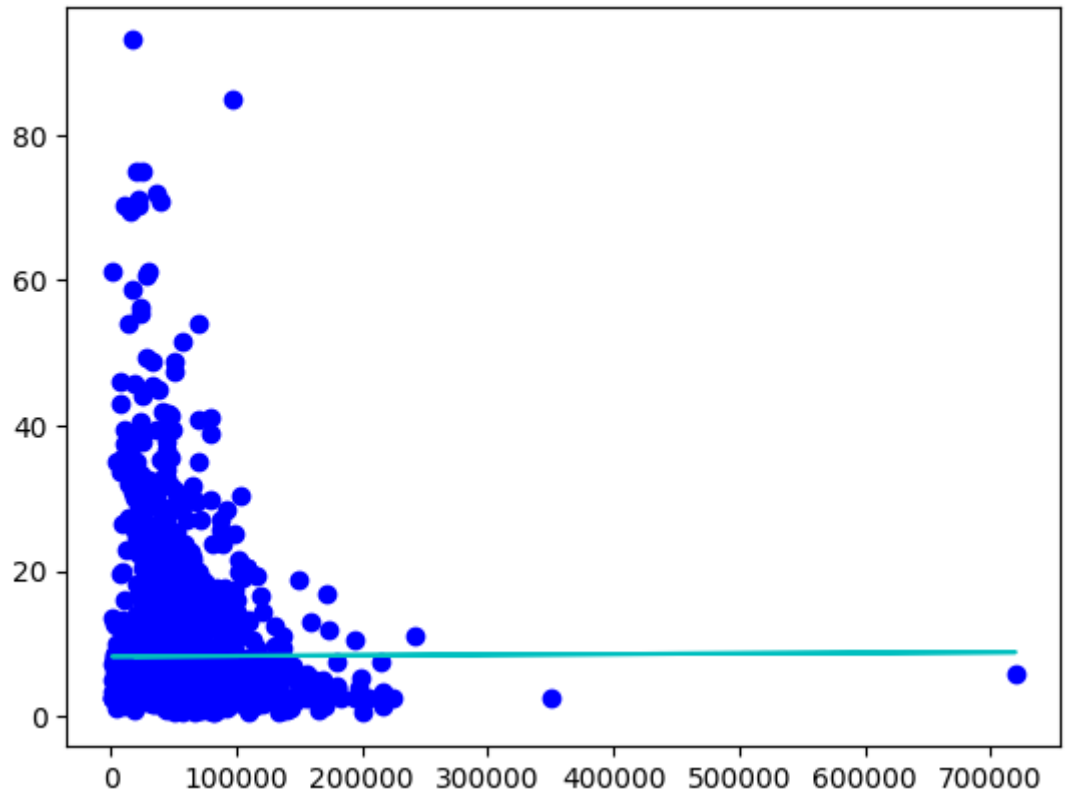
```
-0.0009939448569544762
```

In [15]:

```
y_pred = regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='c')
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x201b6e012d0>]
```



In [16]:

```
df200 = df[:][:200]
df200
```

Out[16]:

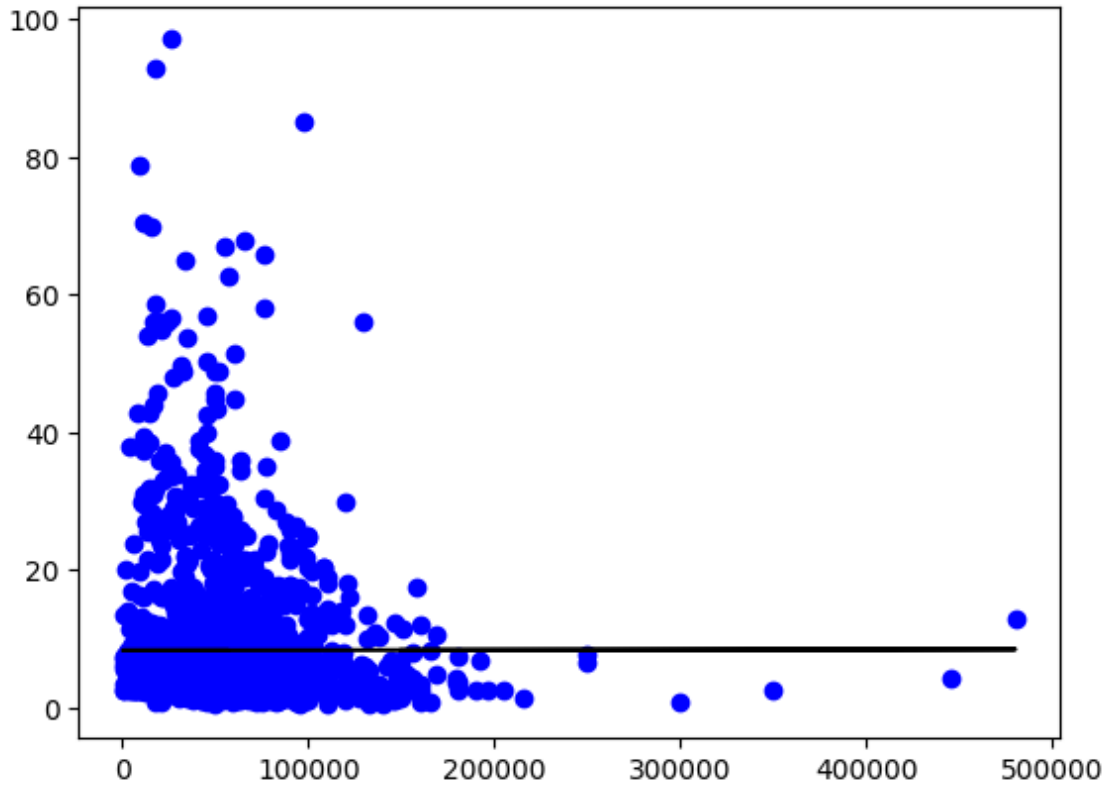
	Km	Cost
0	72000	1.75
1	41000	12.50
2	46000	4.50
3	87000	6.00
4	40670	17.74
...	...	...
195	52000	3.50
196	43571	3.55
197	50000	3.25
198	113000	4.50
199	90000	5.35

200 rows × 2 columns

In [17]:

```
df200.fillna(method='ffill',inplace=True)
X=np.array(df['Km']).reshape(-1,1)
y=np.array(df['Cost']).reshape(-1,1)
df200.dropna(inplace=True)
X_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.25)
regr=LinearRegression()
regr.fit(X_train,y_train)
print("Regressin: ",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regressin:  -0.0017436033723474686
```



In [18]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(x_test)
r2=r2_score(y_test,y_pred)
print('R2 score: ',r2)
```

```
R2 score:  -0.0017436033723474686
```

In [19]:

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(x_test)
r2=metrics.mean_squared_error(y_test,y_pred)
print('MSE: ',r2)

```

```
MSE:  104.74884224431655
```

Logistic Regression

In [20]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

```

In [21]:

```
a=pd.read_csv(r"C:\Users\HP\OneDrive\Documents\used_cars_data-2.csv")
a
```

Out[21]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Mil
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	k
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	1
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	2
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	
...	...	...	...	...	...	...	...	
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	2
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	1
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	2
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	

7253 rows × 14 columns

In [22]:

```
1 a.describe()
```

Out[22]:

	S.No.	Year	Kilometers_Driven	Seats	Price
count	7253.000000	7253.000000	7.253000e+03	7200.000000	6019.000000
mean	3626.000000	2013.365366	5.869906e+04	5.279722	9.479468
std	2093.905084	3.254421	8.442772e+04	0.811660	11.187917
min	0.000000	1996.000000	1.710000e+02	0.000000	0.440000
25%	1813.000000	2011.000000	3.400000e+04	5.000000	3.500000
50%	3626.000000	2014.000000	5.341600e+04	5.000000	5.640000
75%	5439.000000	2016.000000	7.300000e+04	5.000000	9.950000
max	7252.000000	2019.000000	6.500000e+06	10.000000	160.000000

In [23]:

```
1 a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   S.No.              7253 non-null   int64  
 1   Name               7253 non-null   object 
 2   Location           7253 non-null   object 
 3   Year               7253 non-null   int64  
 4   Kilometers_Driven  7253 non-null   int64  
 5   Fuel_Type          7253 non-null   object 
 6   Transmission       7253 non-null   object 
 7   Mileage            7251 non-null   object 
 8   Engine             7207 non-null   object 
 9   Power              7207 non-null   object 
 10  Seats              7200 non-null   float64
 11  New_Price          1006 non-null   object 
 12  Price              6019 non-null   float64
 13  Owner_Type         7253 non-null   object 
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

In [24]:

```
1 a.isnull().sum()
```

Out[24]:

```
S.No.                  0
Name                   0
Location               0
Year                   0
Kilometers_Driven      0
Fuel_Type              0
Transmission           0
Mileage                2
Engine                46
Power                 46
Seats                 53
New_Price           6247
Price               1234
Owner_Type             0
dtype: int64
```

In [25]:

```
1 a.fillna(method='ffill',inplace=True)
```

In [26]:

```
1 a.dropna(inplace=True)
```

In [27]:

```
1 a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7251 entries, 2 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   S.No.              7251 non-null   int64  
 1   Name               7251 non-null   object 
 2   Location           7251 non-null   object 
 3   Year               7251 non-null   int64  
 4   Kilometers_Driven  7251 non-null   int64  
 5   Fuel_Type          7251 non-null   object 
 6   Transmission       7251 non-null   object 
 7   Mileage            7251 non-null   object 
 8   Engine             7251 non-null   object 
 9   Power              7251 non-null   object 
 10  Seats              7251 non-null   float64
 11  New_Price          7251 non-null   object 
 12  Price              7251 non-null   float64
 13  Owner_Type         7251 non-null   object 
dtypes: float64(2), int64(3), object(9)
memory usage: 849.7+ KB
```

In [28]:

```
1 a.isnull().sum()
```

Out[28]:

```
S.No.                0
Name                 0
Location             0
Year                 0
Kilometers_Driven    0
Fuel_Type            0
Transmission         0
Mileage              0
Engine               0
Power                0
Seats                0
New_Price            0
Price                0
Owner_Type           0
dtype: int64
```

In [29]:

```
1 print("This DataFrame has %d rows and %d columns"%(a.shape))
```

```
This DataFrame has 7251 rows and 14 columns
```

In [30]:

```
1 a.head()
```

Out[30]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Mileage
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	18.2 kmpl
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	20.77 kmpl
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	15.2 kmpl
5	5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	LPG	Manual	21.1 km/kg
6	6	Nissan Micra Diesel XV	Jaipur	2013	86999	Diesel	Manual	23.08 kmpl

In [31]:

```
feature_matrix = a.iloc[:,0:13]
target_vector = a.iloc[:,-1]
```

In [32]:

```
print("The feature_matrix has %d rows and %d columns"%(feature_matrix.shape))
```

```
The feature_matrix has 7251 rows and 13 columns
```

In [33]:

```
print("The target_vector has %d rows and %d columns"%(np.array(target_vector).resha
```

```
The target_vector has 7251 rows and 1 columns
```

In [45]:

```
1 feature_matrix_standardized = StandardScaler().fit_transform(feature_matrix)
```

```
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_8628\2404365138.py in ?()
----> 1 feature_matrix_standardized = StandardScaler().fit_transform(feat
ure_matrix)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils
\_set_output.py in ?(self, X, *args, **kwargs)
    138     @wraps(f)
    139     def wrapped(self, X, *args, **kwargs):
--> 140         data_to_wrap = f(self, X, *args, **kwargs)
    141         if isinstance(data_to_wrap, tuple):
    142             # only wrap the first output for cross decomposition
    143             return (

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.
py in ?(self, X, y, **fit_params)
    874         # non-optimized default implementation; override when a b
etter
    875         # method is possible for a given clustering algorithm
    876         if y is None:
    877             # fit method of arity 1 (unsupervised transformation)
--> 878             return self.fit(X, **fit_params).transform(X)
    879         else:
    880             # fit method of arity 2 (supervised transformation)
    881             return self.fit(X, y, **fit_params).transform(X)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\prepr
ocessing\_data.py in ?(self, X, y, sample_weight)
    820             Fitted scaler.
    821         """
    822         # Reset internal state before fitting
    823         self._reset()
--> 824         return self.partial_fit(X, y, sample_weight)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\prepr
ocessing\_data.py in ?(self, X, y, sample_weight)
    857         """
    858         self._validate_params()
    859
    860         first_call = not hasattr(self, "n_samples_seen_")
--> 861         X = self._validate_data(
    862             X,
    863             accept_sparse=("csr", "csc"),
    864             dtype=FLOAT_DTYPES,

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.
py in ?(self, X, y, reset, validate_separately, **check_params)
    561
    562         if no_val_X and no_val_y:
    563             raise ValueError("Validation should be done on X, y o
r both.")
    564         elif not no_val_X and no_val_y:
--> 565             X = check_array(X, input_name="X", **check_params)
    566             out = X
    567         elif no_val_X and not no_val_y:
    568             y = _check_y(y, **check_params)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils
```

```
\validation.py in ?(array, accept_sparse, accept_large_sparse, dtype, ord
er, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensu
re_min_features, estimator, input_name)
    876                         )
    877                     array = xp.astype(array, dtype, copy=False)
    878                 else:
    879                     array = _asarray_with_order(array, order=orde
r, dtype=dtype, xp=xp)
--> 880             except ComplexWarning as complex_warning:
    881                 raise ValueError(
    882                     "Complex data not supported\n{}\n".format(arr
ay)
    883                 ) from complex_warning

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils
\_array_api.py in ?(array, dtype, order, copy, xp)
    181     if xp is None:
    182         xp, _ = get_namespace(array)
    183     if xp.__name__ in {"numpy", "numpy.array_api"}:
    184         # Use NumPy API to support order
--> 185         array = numpy.asarray(array, order=order, dtype=dtype)
    186         return xp.asarray(array, copy=copy)
    187     else:
    188         return xp.asarray(array, dtype=dtype, copy=copy)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\g
eneric.py in ?(self, dtype)
   1996     def __array__(self, dtype: npt.DTypeLike | None = None) -> n
p.ndarray:
   1997         values = self._values
-> 1998         arr = np.asarray(values, dtype=dtype)
   1999         if (
   2000             astype_is_view(values.dtype, arr.dtype)
   2001             and using_copy_on_write()

ValueError: could not convert string to float: 'Honda Jazz V'
```

In [46]:

```
1 algorithm = LogisticRegression(penalty ='l2',dual=False,tol=1e-4,C=1.0,fit_intercep
```

In [47]:

```
1 Logistic_Regression_Model = algorithm.fit(feature_matrix_standardized,target_vector
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Cell In[47], line 1
----> 1 Logistic_Regression_Model = algorithm.fit(feature_matrix_standard
ized,target_vector)

NameError: name 'feature_matrix_standardized' is not defined
```

In [48]:

```
observation = [[1,0,0.9,0.8000,9.789,9.086,0.956,0.0005,0.675,0.846,0.82541,0.94673
```

In [49]:

```
predictions = Logistic_Regression_Model.predict(observation)
print('The Model predicted The observation to belong to class %s'%(predictions))
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Cell In[49], line 1
----> 1 predictions = Logistic_Regression_Model.predict(observation)
      2 print('The Model predicted The observation to belong to class %
s'%(predictions))

NameError: name 'Logistic_Regression_Model' is not defined
```

In [50]:

```
print('Algorithm was Trained To predict one of the two classes : %s'%(algorithm.cla
```

```
---------------------------------------------------------------------------
--
AttributeError                            Traceback (most recent call las
t)
Cell In[50], line 1
----> 1 print('Algorithm was Trained To predict one of the two classes :
%s'%(algorithm.classes_))

AttributeError: 'LogisticRegression' object has no attribute 'classes_'
```

In [51]:

```
1 print("""The Model ssays the probability of the observation we passed Belonging To (
2 print()
3 print("""The Model ssays the probability of the observation we passed Belonging To (
```

```
---------------------------------------------------------------------
--
NotFittedError                            Traceback (most recent call las
t)
Cell In[51], line 1
----> 1 print("""The Model ssays the probability of the observation we pa
ssed Belonging To class['No'] Is %s"""%(algorithm.predict_proba(observati
on)[0][0]))
      2 print()
      3 print("""The Model ssays the probability of the observation we pa
ssed Belonging To class['Yes'] Is %s"""%(algorithm.predict_proba(observat
ion)[0][1]))

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\linear_model\_logistic.py:1362, in LogisticRegression.predict_proba(sel
f, X)
   1336 def predict_proba(self, X):
   1337     """
   1338     Probability estimates.
   1339
   (...)
   1360         where classes are ordered as they are in ``self.classes_``
``.
   1361     """
-> 1362     check_is_fitted(self)
   1364     ovr = self.multi_class in ["ovr", "warn"] or (
   1365         self.multi_class == "auto"
   1366         and (
   (...)
   1369         )
   1370     )
   1371     if ovr:

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:1390, in check_is_fitted(estimator, attributes, msg,
all_or_any)
   1385     fitted = [
   1386         v for v in vars(estimator) if v.endswith("_") and not v.s
tartswith("__")
   1387     ]
   1389 if not fitted:
-> 1390     raise NotFittedError(msg % {"name": type(estimator).__name_
_})

NotFittedError: This LogisticRegression instance is not fitted yet. Call
'fit' with appropriate arguments before using this estimator.
```

In [ ]:

```
1
```