APPPLIED DATA SCINECE PROJECT

PROJECT TITLE:

PREDICTION OF CO2 EMISSION IN VEHICLES

TEAM MEMEBERS - 12

1. **Ch.V.Abhijeet**
2. **Mohammed Afzal**
3. **M Venkat Anvay Reddy**
4. **Vikas Yadav**

1. Introduction

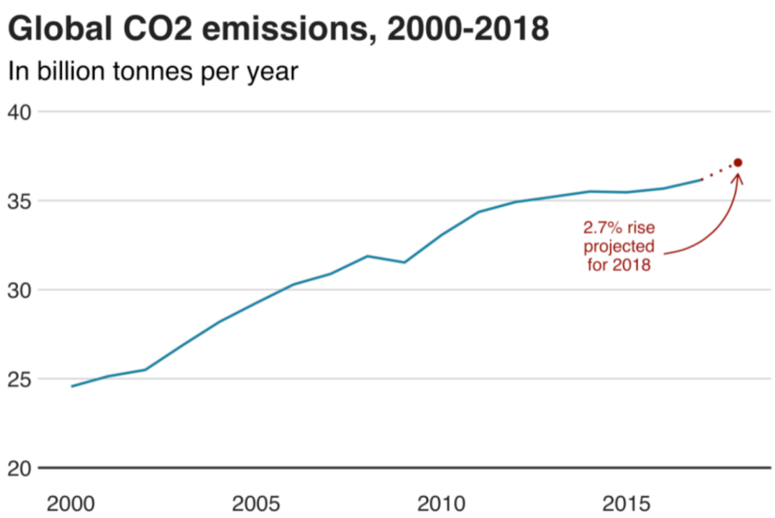
Overview: The project was to predict the emission of CO_2 in vehicles this dataset contains official CO_2 emissions data for various cars of different features over the period of 2014 to 2020. It has 7385 samples and has a total of 11 features.

Purpose: The main purpose of the project was to find the rate of CO_2 emission in vehicles and to find which type of cars and depending on the fuel consumption which cars emit the less CO_2.

2. Literature Survey

Existing Problem: Our personal vehicles are a major cause of global warming, collectively cars and trucks account for nearly one-fifth of emissions, emitting around 24 pounds of carbon dioxide and other global-warming gases for every gallon of gas.

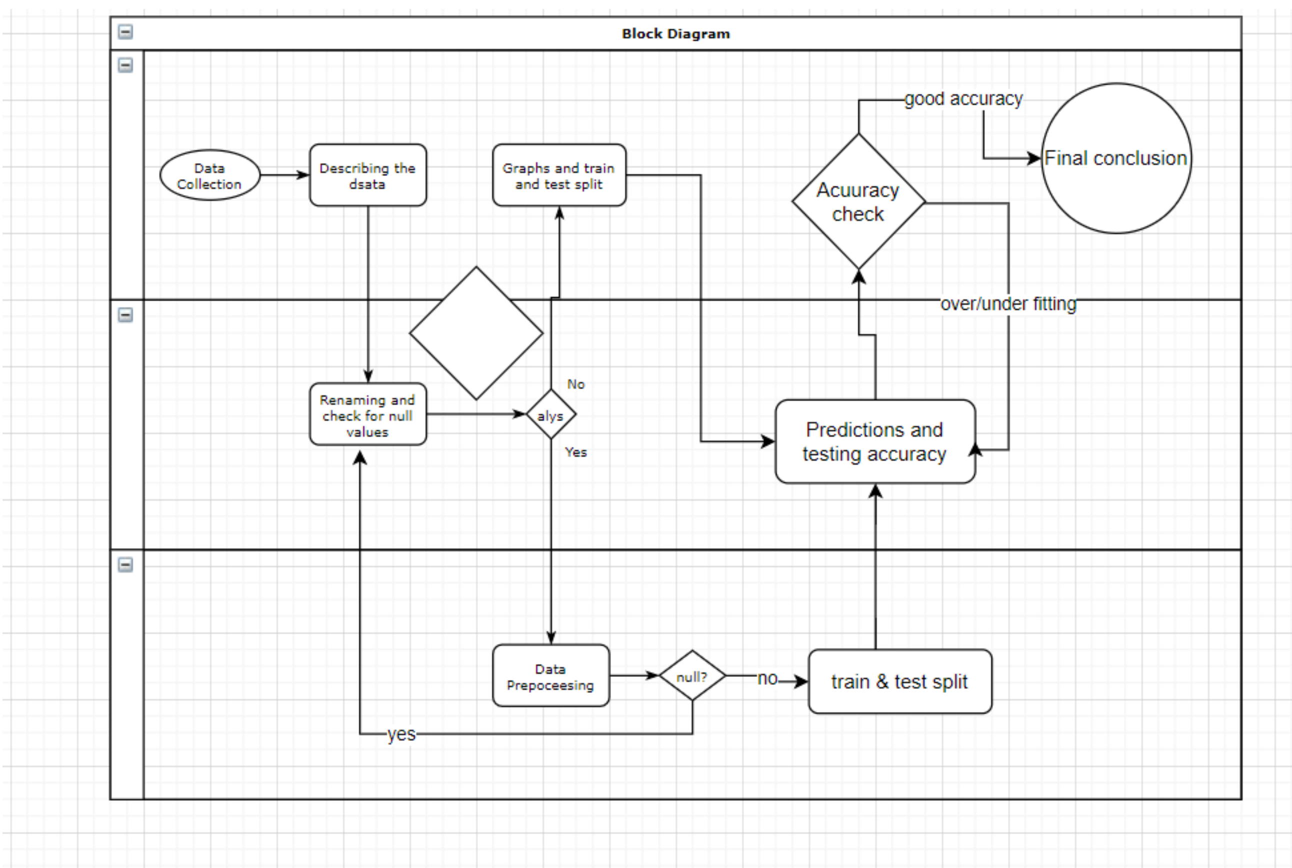
Most of the gases comes from the extraction, production, and delivery of the fuel, while the great bulk of heat-trapping emissions more than 19 pounds per gallon comes right out of a car's tailpipe.



Proposed Solution: The idea is to take the data from the past 10 years and the data set includes different kinds of vehicles and their CO_2 emissions based on the fuel consumption and if we are able to calculate the vehicles which produce more CO_2 as compared to other vehicles which have same engine size and fuel consumption we can decrease or stop the production of such vehicles and based on the predictions we can come to form a formidable conclusion on what to do.

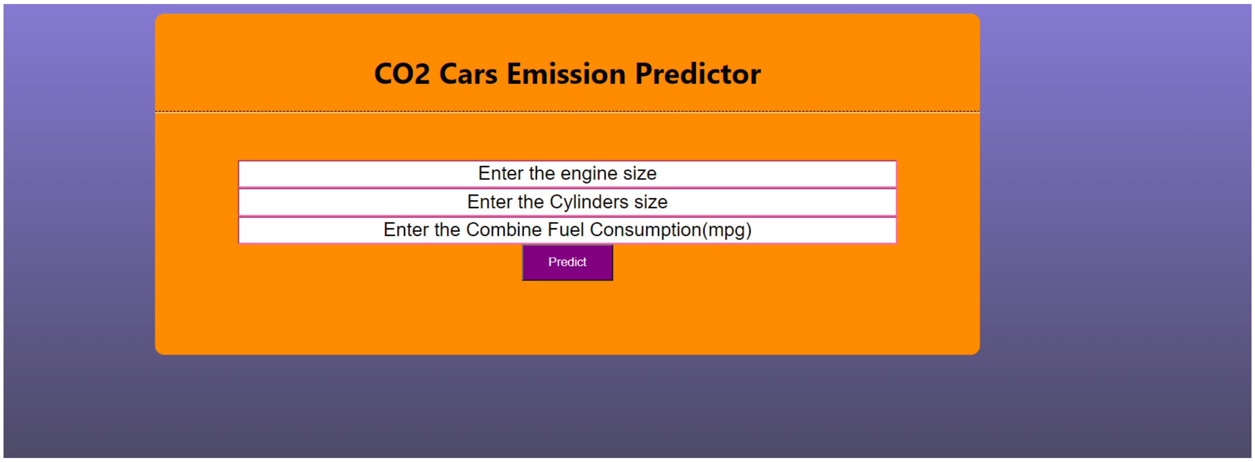
3. Theoretical Analysis

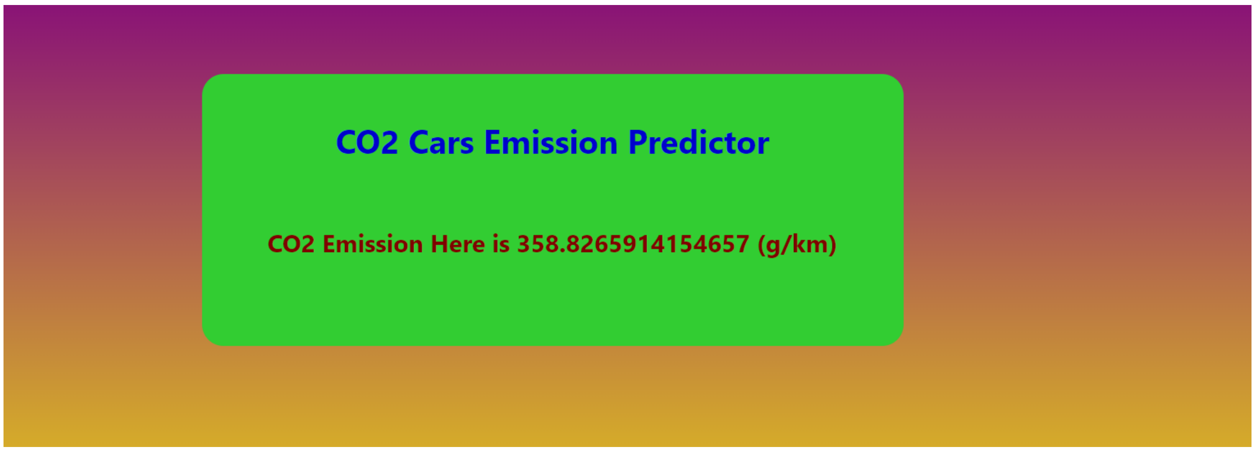
Block Diagram:



Hardware/ Software Designing:

We designed a web page which is used to predict the Co2 based on Combine Fuel Consumption and engine, cylinder size.





The hardware components required for the project are:

8GB RAM

The software components required for the project are:

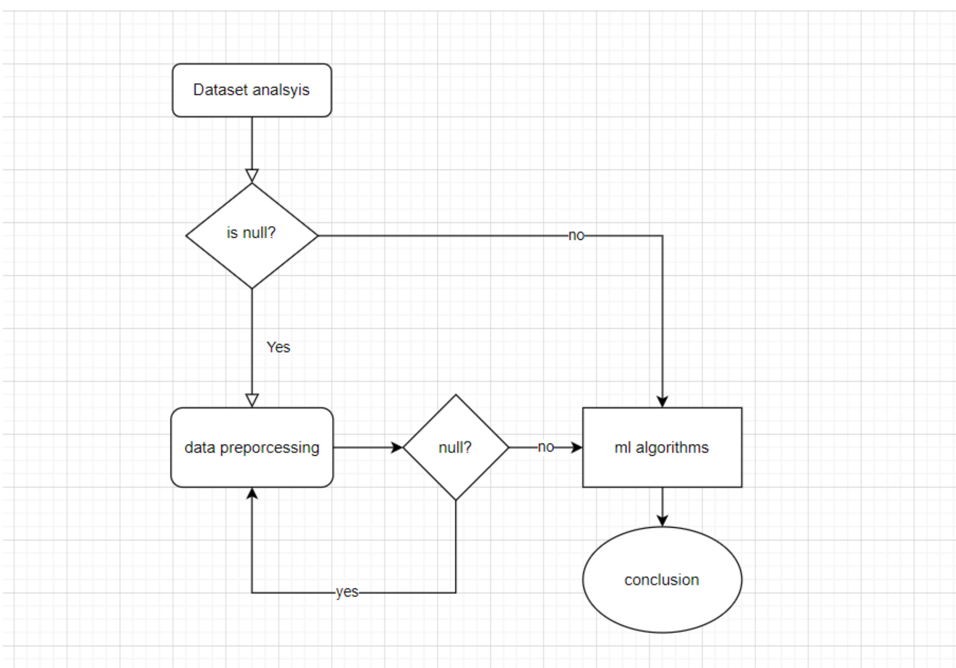
Anaconda prompt

Spyder

4. Experimental Investigations

During the project analysis we first observed what kind of data we had and as there were few empty spaces in the names we replaced the names values and checked for null values if there are any and we preprocessed the data and made few more observations and calculated the correlation and based on that we extracted the columns we need and then train and test splitting of data was done we applied different algorithms and based on the accuracy prediction we came to a conclusion that Linear regression gives good accuracy.

5. Flowchart



6. Result

Based on the algorithms we applies we came the conclusion that linear regression is giving good predictions compared to other algorithms which were either under or over fitting.

7. Advantages

The main advantage of this project is that we will be able to predict which cars are environment supportive and which cars emit less CO_2 .

Disadvantages

Though there are certain advantages we may not be able to predict which car gives consumes less CO_2 though the accuracy is 86 we might not be able to give exact accuracy which may be improved if we collect a much larger dataset and run it in a data warehouse.

8. Applications

- They may be useful for the production of vehicles which emit less CO_2.
- They will be helpful for the better development of engine size.
- They may be useful for checking the ration of engine to fuel consumption which emit less CO_2.

9. Conclusion

We came to a final conclusion that engine size , cylinder size and fuel consumption directly affect the amount of CO_2 which is being released into the atmosphere.

10. Future Scope

If we are able to calculate the collect much larger and amount of dataset and test the engine components for various items and how much fuel is being consumed by them and if we run the dataset in a **DATAWAREHOUSE** and deploy a machine learning model and algorithms then we may improve accuracy and give better results.

11. Bibliography

- https://www.kaggle.com/debajyotipodder/basic-eda-of-the-co2-emissions-by-vehicle-dataset
- https://www.google.com/search?q=increase+in+co2+emission+d%3Due+to+cars&sxsrf=ALeKk02YNk-M1K4CbYgCAeix1TesXB-8BA:1627795169822&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi1zPTGiY_yAhWIA3IKHSsBD_EQ_AUoAnoECAEQBA&biw=1536&bih=722#imgrc=Fx1zQE03huyuhM
- https://www.ucsusa.org/resources/car-emissions-global-warming
- https://sso.teachable.com/secure/teachable_accounts

12. Appendix

Source Code

Renaming the Columns

In [3]:
```
renamed_col = {
    'Vehicle Class': 'vehicle_class',
    'Engine Size(L)': 'engine_size',
    'Fuel Type': 'fuel_type',
    'Fuel Consumption City (L/100 km)': 'fuel_cons_city',
    'Fuel Consumption Hwy (L/100 km)': 'fuel_cons_hwy',
    'Fuel Consumption Comb (L/100 km)': 'fuel_cons_comb',
    'Fuel Consumption Comb (mpg)': 'mpgfuel_cons_comb',
    'CO2 Emissions(g/km)': 'co2'
}
dataset.rename(renamed_col, axis='columns', inplace=True)
```

Checking For Null Values

In [4]:
```
dataset.isnull().any()
```

Out[4]:
```
Make                 False
Model                False
vehicle_class        False
engine_size          False
Cylinders            False
Transmission         False
fuel_type            False
fuel_cons_city       False
fuel_cons_hwy        False
fuel_cons_comb       False
mpgfuel_cons_comb    False
co2                  False
dtype: bool
```

In [5]:
```
dataset.isnull().sum()
```

Out[5]:
```
Make                 0
Model                0
vehicle_class        0
engine_size          0
Cylinders            0
Transmission         0
fuel_type            0
fuel_cons_city       0
fuel_cons_hwy        0
fuel_cons_comb       0
mpgfuel_cons_comb    0
co2                  0
dtype: int64
```

In [6]:
```
dataset.columns
```

Out[6]:
```
Index(['Make', 'Model', 'vehicle_class', 'engine_size', 'Cylinders',
       'Transmission', 'fuel_type', 'fuel_cons_city', 'fuel_cons_hwy',
       'fuel_cons_comb', 'mpgfuel_cons_comb', 'co2'],
      dtype='object')
```

Information of Dataset

In [7]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Make               7385 non-null   object
 1   Model              7385 non-null   object
 2   vehicle_class      7385 non-null   object
 3   engine_size        7385 non-null   float64
 4   Cylinders          7385 non-null   int64
 5   Transmission       7385 non-null   object
 6   fuel_type          7385 non-null   object
 7   fuel_cons_city     7385 non-null   float64
 8   fuel_cons_hwy      7385 non-null   float64
 9   fuel_cons_comb     7385 non-null   float64
 10  mpgfuel_cons_comb  7385 non-null   int64
 11  co2                7385 non-null   int64
dtypes: float64(4), int64(3), object(5)
memory usage: 692.5+ KB
```

In [8]:
```
dataset.describe()
```

Out[8]:

	engine_size	Cylinders	fuel_cons_city	fuel_cons_hwy	fuel_cons_comb	mpgfuel_cons_comb	co2
count	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000
mean	3.160068	5.615030	12.556534	9.041706	10.975071	27.481652	250.584699
std	1.354170	1.828307	3.500274	2.224456	2.892506	7.231879	58.512679
min	0.900000	3.000000	4.200000	4.000000	4.100000	11.000000	96.000000
25%	2.000000	4.000000	10.100000	7.500000	8.900000	22.000000	208.000000
50%	3.000000	6.000000	12.100000	8.700000	10.600000	27.000000	246.000000
75%	3.700000	6.000000	14.600000	10.200000	12.600000	32.000000	288.000000
max	8.400000	16.000000	30.600000	20.600000	26.100000	69.000000	522.000000

Data Visualization

In [9]:
```
dataset.Make.unique()
```

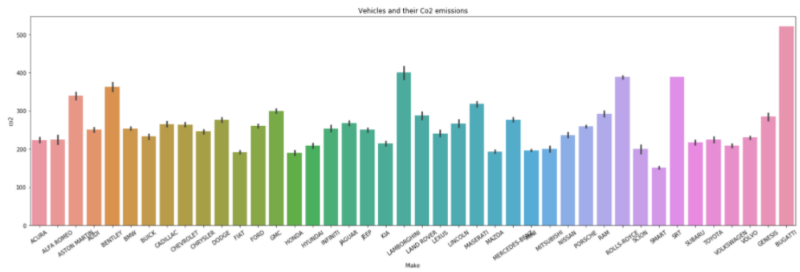
Out[9]:
```
array(['ACURA', 'ALFA ROMEO', 'ASTON MARTIN', 'AUDI', 'BENTLEY', 'BMW',
       'BUICK', 'CADILLAC', 'CHEVROLET', 'CHRYSLER', 'DODGE', 'FIAT',
       'FORD', 'GMC', 'HONDA', 'HYUNDAI', 'INFINITI', 'JAGUAR', 'JEEP',
       'KIA', 'LAMBORGHINI', 'LAND ROVER', 'LEXUS', 'LINCOLN', 'MASERATI',
       'MAZDA', 'MERCEDES-BENZ', 'MINI', 'MITSUBISHI', 'NISSAN',
       'PORSCHE', 'RAM', 'ROLLS-ROYCE', 'SCION', 'SMART', 'SRT', 'SUBARU',
       'TOYOTA', 'VOLKSWAGEN', 'VOLVO', 'GENESIS', 'BUGATTI'],
      dtype=object)
```

Make VS CO2 Emission

In [10]:
```
f, ax = plt.subplots(figsize=(25,7))

x = dataset.Make.value_counts().sort_values()

ax = sns.barplot(data=dataset,x='Make',y='co2')
plt.title('Vehicles and their Co2 emissions')
plt.xticks(rotation=35)
plt.show()
```



Model Building Using Decision Tree

In [52]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
```

In [53]:
```
dt_model=DecisionTreeRegressor(criterion='mse',random_state=0)
```

In [54]:
```
dt_model.fit(x_train,y_train)
```

Out[54]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', 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='deprecated',
                      random_state=0, splitter='best')
```

In [55]:
```
y_pred=dt_model.predict(x_test)
y_pred
```

Out[55]:
```
array([300.        , 176.25      , 286.25      , ..., 246.76190476,
       323.55555556, 250.5625    ])
```

In [56]:
```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[56]:
```
7.420156024586886
```

In [57]:
```
dt_params = {'max_depth': list(range(1,10)),
            'max_features': [3,5,10,15]}
```

In [58]:
```
dt_model = DecisionTreeRegressor(random_state = 42)
```

Using GridSearch for hyperparameter tuning

In [59]:
```
dt_cv_model = GridSearchCV(dt_model,
                           dt_params,
                           cv = 10,
                            n_jobs = -1,
                          verbose = 2)
```

In [60]:
```
dt_cv_model.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 36 candidates, totalling 360 fits
```

Scaling

```
In [39]: from sklearn.preprocessing import StandardScaler
         import joblib
         sc=StandardScaler()
         x=sc.fit_transform(x)
         x
         joblib.dump(sc,'standard.save')

Out[39]: ['standard.save']
```

Splitting the Dataset into train and test

```
In [40]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Model Building Using Multiple Linear Regression

```
In [41]: from sklearn.linear_model import LinearRegression

In [42]: mr=LinearRegression()
```

Fitting the model to training data

```
In [43]: model=mr.fit(x_train,y_train)
```

Making Predictions

```
In [44]: y_pred=mr.predict(x_test)
         y_pred

Out[44]: array([373.41938679, 172.7802348 , 272.18033419, ..., 241.51011101,
                321.57600975, 233.59796301])

In [45]: y_test

Out[45]: 553     290
```

```
In [60]: dt_cv_model.fit(x_train, y_train)

         Fitting 10 folds for each of 36 candidates, totalling 360 fits

         [Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
         [Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    4.5s
         [Parallel(n_jobs=-1)]: Done 322 tasks      | elapsed:    5.3s
         [Parallel(n_jobs=-1)]: Done 360 out of 360 | elapsed:    5.4s finished

Out[60]: GridSearchCV(cv=10, error_score=nan,
                      estimator=DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse',
                                                      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='deprecated',
                                                      random_state=42, splitter='best'),
                      iid='deprecated', n_jobs=-1,
                      param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                                  'max_features': [3, 5, 10, 15]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=2)

In [61]: dt_cv_model.best_params_

Out[61]: {'max_depth': 9, 'max_features': 3}

In [62]: dt_tuned = DecisionTreeRegressor(max_depth  = 9,
                                          max_features = 3)

In [63]: dt_tuned.fit(x_train, y_train)

Out[63]: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=9,
                               max_features=3, 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='deprecated',
                               random_state=None, splitter='best')

In [64]: np.sqrt(mean_squared_error(y_test, y_pred))

Out[64]: 7.420156024586886
```

Checking the Score of The Model

In [65]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[65]: 0.9814724059217347

In [66]:
```
from io import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(dt_tuned,out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.600777 to fit
```

Out[66]:

In [67]:
```
pred=dt_tuned.predict(x_test)
frames = [pred,y_test]
result_pred = pd.DataFrame(data=frames)
result_pred=result_pred.T

result_pred_Dt=result_pred.rename(columns={0:'Pred_DT',1:'Real'})
result_pred_Dt["Pred_DT"]=result_pred_Dt["Pred_DT"].map(lambda x:round(x,2))
result_pred_Dt["Diff"]=result_pred_Dt["Pred_DT"]-result_pred_Dt["Real"]
result_pred_Dt["Diff"]=result_pred_Dt["Diff"]
print("Mean Diff: ",abs(result_pred_Dt["Diff"]).mean())
result_pred_Dt.head(20)
```

```
Mean Diff:  3.6608868715083807
```

Out[67]:

	Pred_DT	Real	Diff
0	308.02	290.0	18.02
1	176.00	178.0	-2.00
2	285.07	287.0	-1.93
3	272.93	274.0	-1.07
4	186.26	188.0	-1.74
5	209.57	211.0	-1.43

```
Fitting 10 folds for each of 144 candidates, totalling 1440 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   36.0s
[Parallel(n_jobs=-1)]: Done 632 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 997 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 1440 out of 1440 | elapsed:  2.8min finished
```

Out[76]:
```
GridSearchCV(cv=10, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=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=None,
                                             oob_score=False, random_state=42,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                         'max_features': [3, 5, 10, 15],
                         'n_estimators': [100, 200, 500, 750]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=2)
```

In [77]:
```
rf_cv_model.best_params_
```

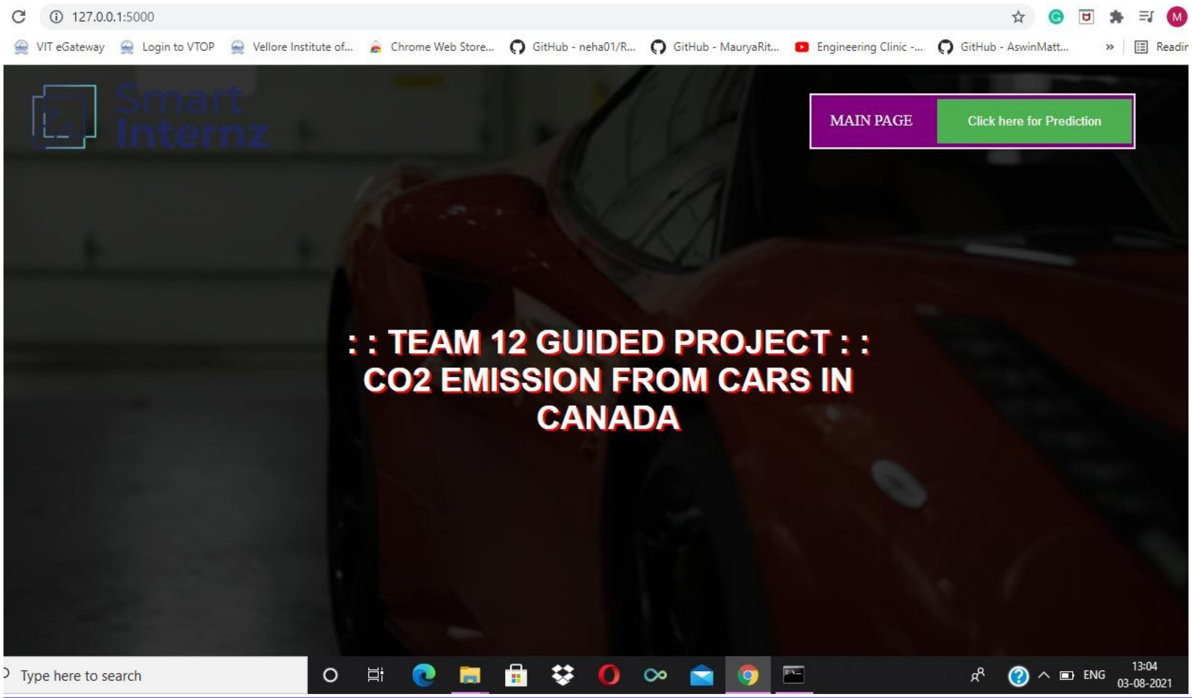
Out[77]: {'max_depth': 9, 'max_features': 3, 'n_estimators': 500}

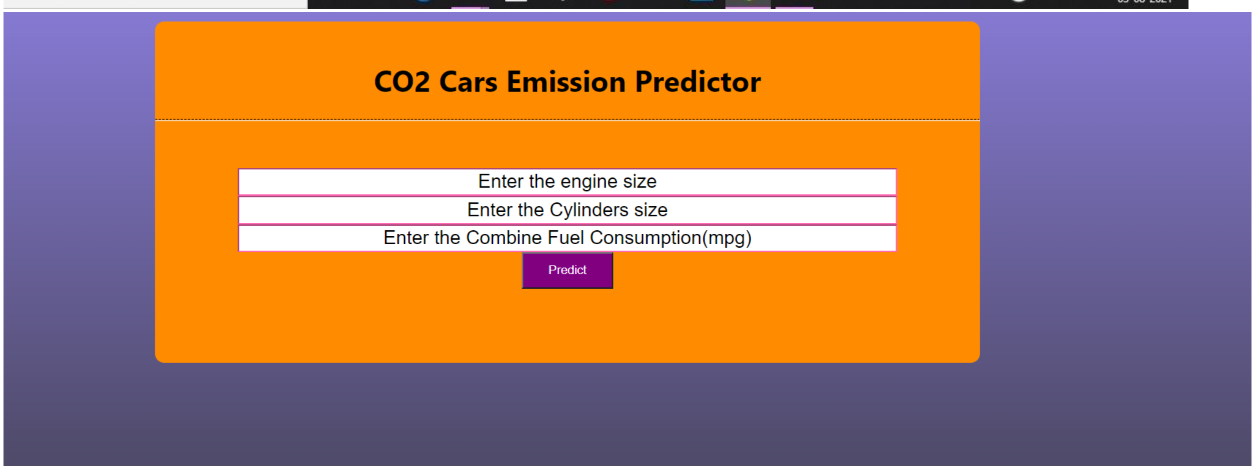
In [78]:
```
rf_tuned = RandomForestRegressor(max_depth  = 9,
                                 max_features = 3,
                                 n_estimators =500)
```

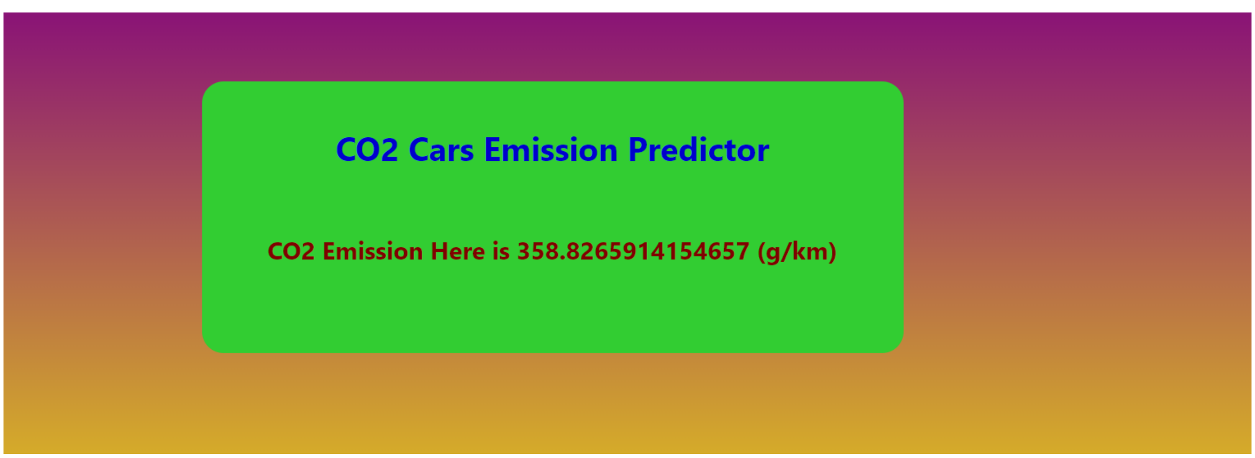
In [79]:
```
rf_tuned.fit(x_train, y_train)
```

Out[79]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=9, max_features=3, max_leaf_nodes=None,
                      max_samples=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=500, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

UI Screenshots:







Editor - A:\Internship\INternship\FINAL PROJECT\finalsite.py

finalsite.py indexnew.html

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 31 14:30:10 2021

@author: M VENKAT ANVAY REDDY
"""

from flask import Flask, request, render_template
import joblib
model = joblib.load('fourtune.save')
trans=joblib.load('scproject.save')

app = Flask(__name__)


@app.route('/')
def home():
    return render_template('home.html')
@app.route('/Prediction',methods=['POST','GET'])
def prediction():
    return render_template('indexnew.html')
@app.route('/Home',methods=['POST','GET'])
def my_home():
    return render_template('home.html')
@app.route('/predict',methods=["POST","GET"])
def predict():
    x_test=[[float(x) for x in request.form.values()]]
    x_test=trans.transform(x_test)
    y_pred=model.predict(x_test)
    output=y_pred[0]
    return render_template('resultnew.html',prediction='{}'.format(output))


if __name__== '__main__':
    app.run(debug=True)
```