Week4: Deployment on Flask

Chaithanya S.I

Data Analytics Batch

LISUM02

Data Glacier
Your Deep Learning Partner

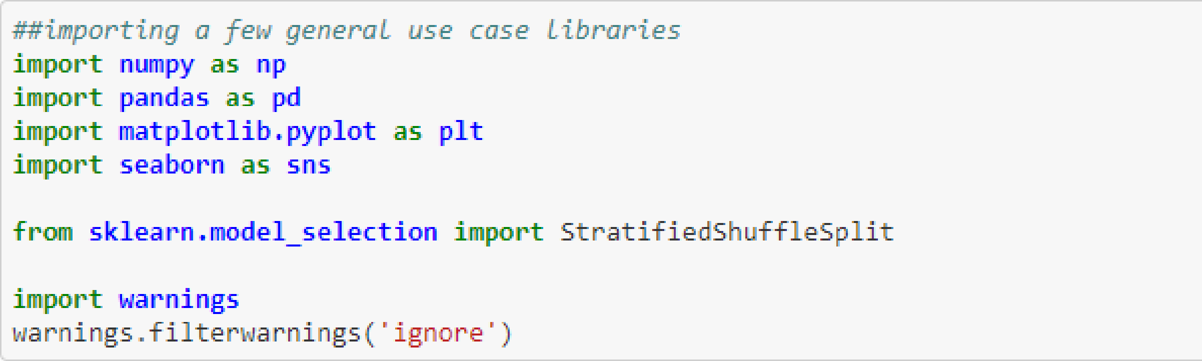
Predicting Fuel Efficiency of Vehicles

1. Selecting dataset (simple data) – http://archive.ics.uci.edu/ml/datasets/Auto+MPG

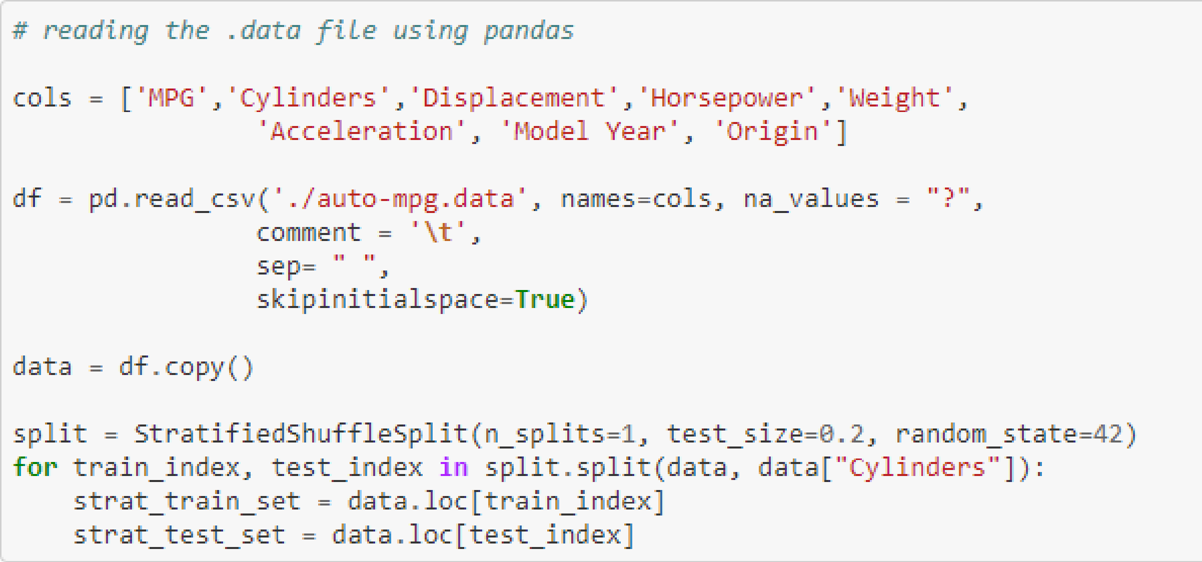
   Attribute Information: Auto MPG Data Set

   1. mpg: continuous
   2. cylinders: multi-valued discrete
   3. displacement: continuous
   4. horsepower: continuous
   5. weight: continuous
   6. acceleration: continuous
   7. model year: multi-valued discrete
   8. origin: multi-valued discrete
   9. car name: string (unique for each instance)

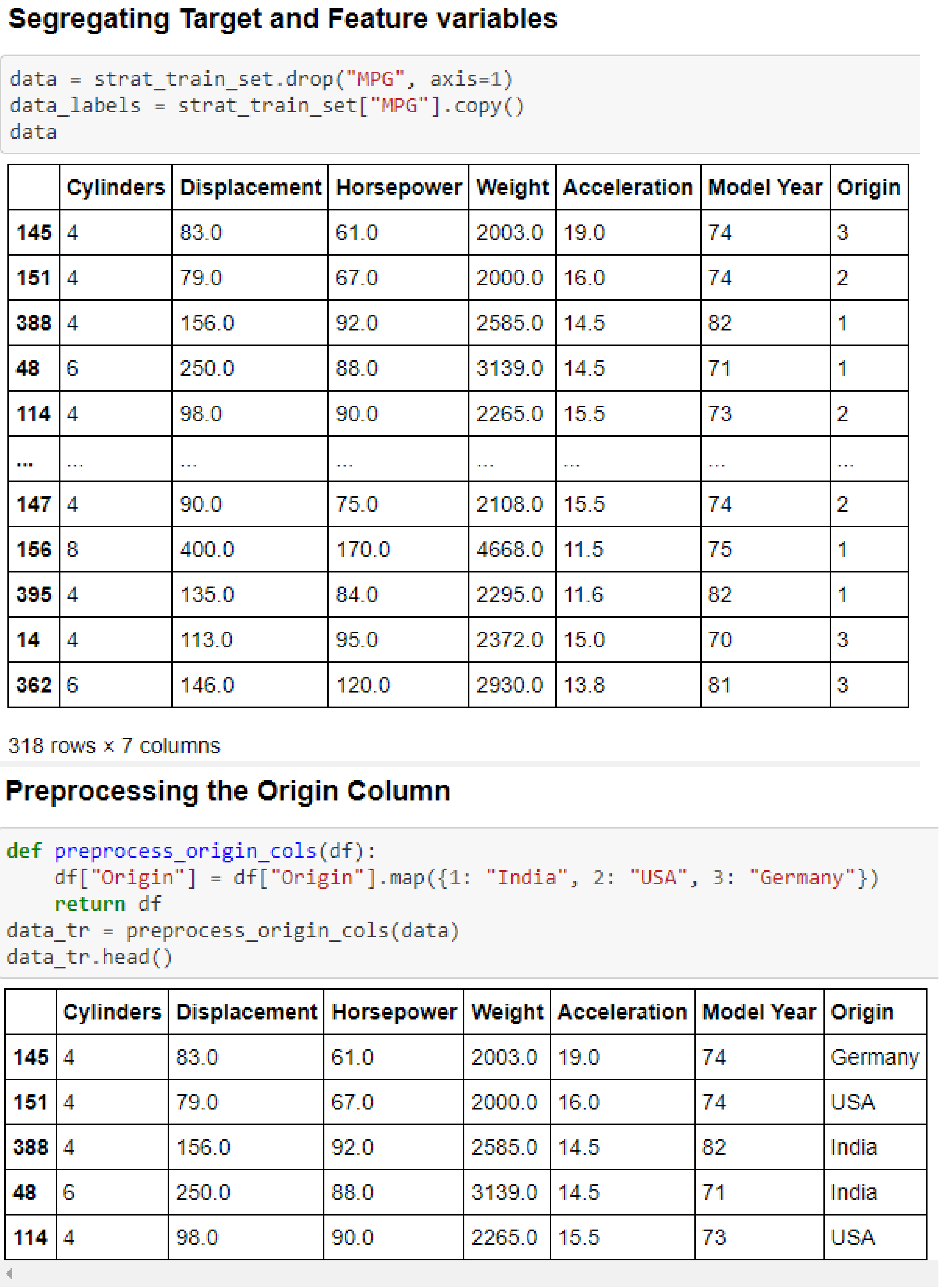
2. Data Preparation



```
##importing a few general use case libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import StratifiedShuffleSplit

import warnings
warnings.filterwarnings('ignore')
```



```
# reading the .data file using pandas

cols = ['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']

df = pd.read_csv('./auto-mpg.data', names=cols, na_values = "?",
                comment = '\t',
                sep= " ",
                skipinitialspace=True)

data = df.copy()

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(data, data["Cylinders"]):
    strat_train_set = data.loc[train_index]
    strat_test_set = data.loc[test_index]
```



Segregating Target and Feature variables

```
data = strat_train_set.drop("MPG", axis=1)
data_labels = strat_train_set["MPG"].copy()
data
```

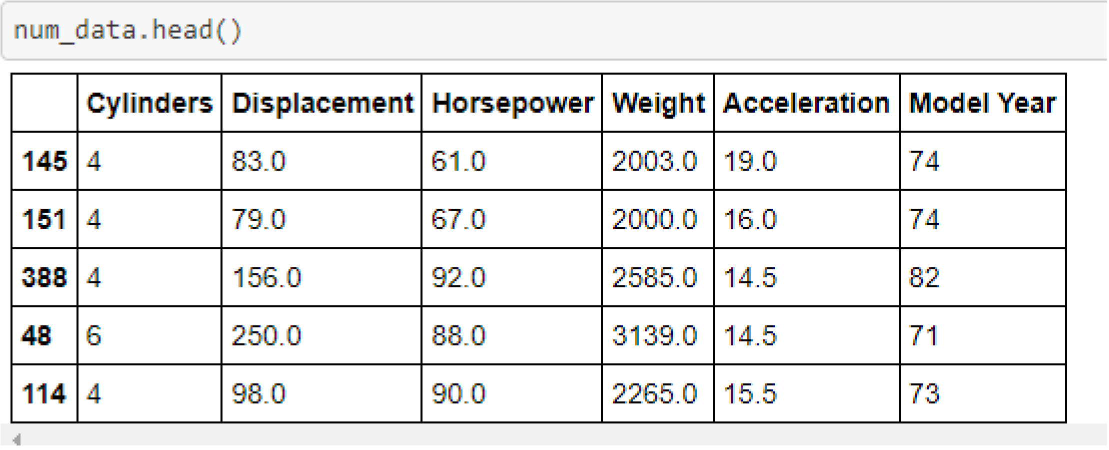
	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
145	4	83.0	61.0	2003.0	19.0	74	3
151	4	79.0	67.0	2000.0	16.0	74	2
388	4	156.0	92.0	2585.0	14.5	82	1
48	6	250.0	88.0	3139.0	14.5	71	1
114	4	98.0	90.0	2265.0	15.5	73	2
...	...	...	...	...	...	...	...
147	4	90.0	75.0	2108.0	15.5	74	2
156	8	400.0	170.0	4668.0	11.5	75	1
395	4	135.0	84.0	2295.0	11.6	82	1
14	4	113.0	95.0	2372.0	15.0	70	3
362	6	146.0	120.0	2930.0	13.8	81	3

318 rows × 7 columns

Preprocessing the Origin Column

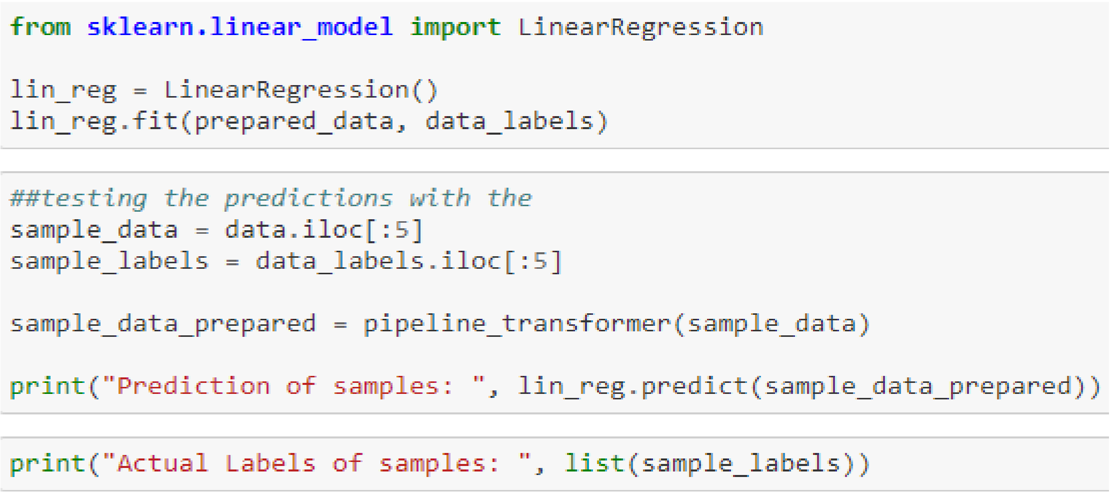
```
def preprocess_origin_cols(df):
    df["Origin"] = df["Origin"].map({1: "India", 2: "USA", 3: "Germany"})
    return df
data_tr = preprocess_origin_cols(data)
data_tr.head()
```

	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
145	4	83.0	61.0	2003.0	19.0	74	Germany
151	4	79.0	67.0	2000.0	16.0	74	USA
388	4	156.0	92.0	2585.0	14.5	82	India
48	6	250.0	88.0	3139.0	14.5	71	India
114	4	98.0	90.0	2265.0	15.5	73	USA



```
num_data.head()
```

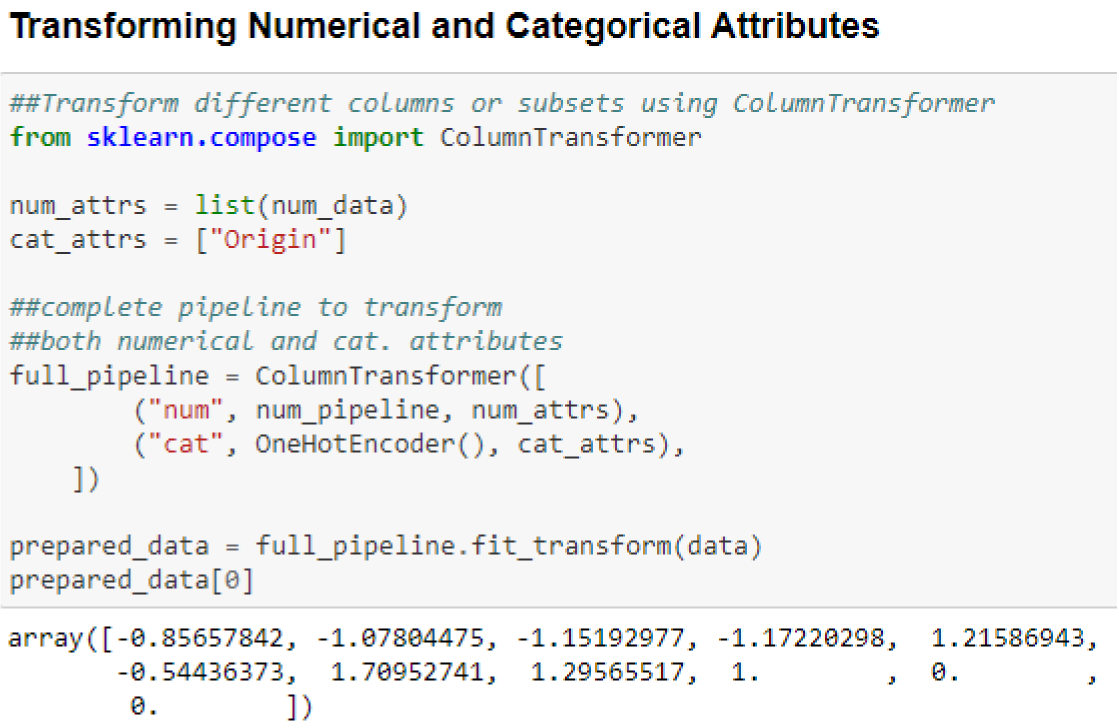
	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year
145	4	83.0	61.0	2003.0	19.0	74
151	4	79.0	67.0	2000.0	16.0	74
388	4	156.0	92.0	2585.0	14.5	82
48	6	250.0	88.0	3139.0	14.5	71
114	4	98.0	90.0	2265.0	15.5	73



```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(prepared_data, data_labels)
```

```
##testing the predictions with the
sample_data = data.iloc[:5]
sample_labels = data_labels.iloc[:5]

sample_data_prepared = pipeline_transformer(sample_data)

print("Prediction of samples: ", lin_reg.predict(sample_data_prepared))
```

```
print("Actual Labels of samples: ", list(sample_labels))
```

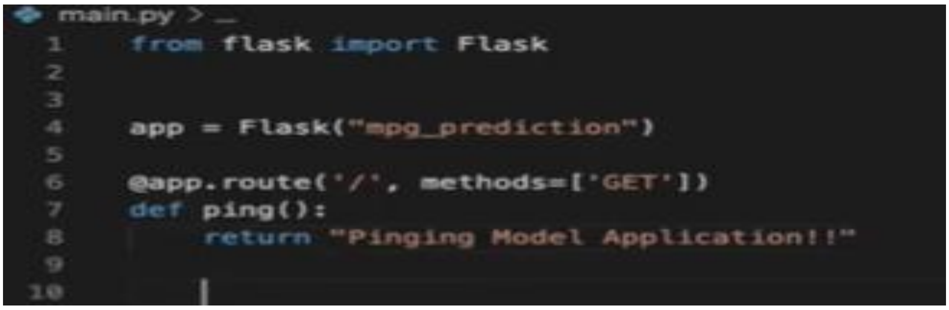


Transforming Numerical and Categorical Attributes

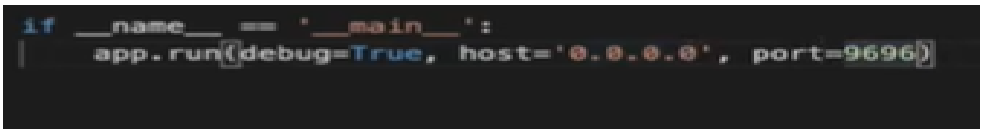
```
##Transform different columns or subsets using ColumnTransformer
from sklearn.compose import ColumnTransformer

num_attrs = list(num_data)
cat_attrs = ["Origin"]

##complete pipeline to transform
##both numerical and cat. attributes
full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attrs),
        ("cat", OneHotEncoder(), cat_attrs),
    ])

prepared_data = full_pipeline.fit_transform(data)
prepared_data[0]
```

```
array([-0.85657842, -1.07804475, -1.15192977, -1.17220298,  1.21586943,
       -0.54436373,  1.70952741,  1.29565517,  1.        ,  0.        ,
        0.        ])
```

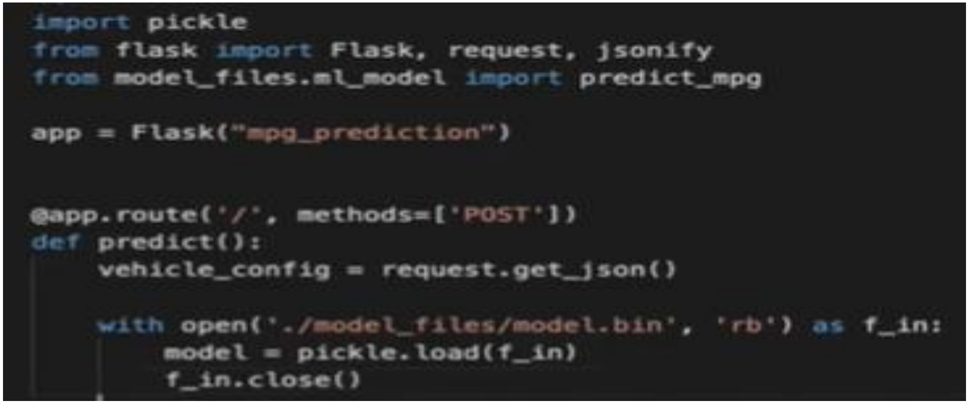
FLASK



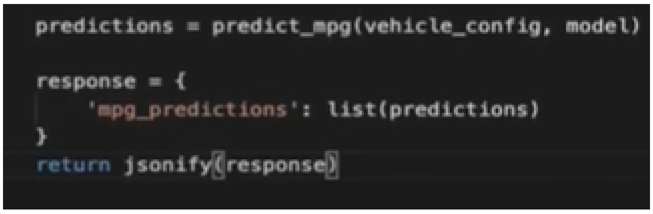
```python
from flask import Flask


app = Flask("mpg_prediction")

@app.route('/', methods=['GET'])
def ping():
    return "Pinging Model Application!!"

    
```



```python
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=9696)
```

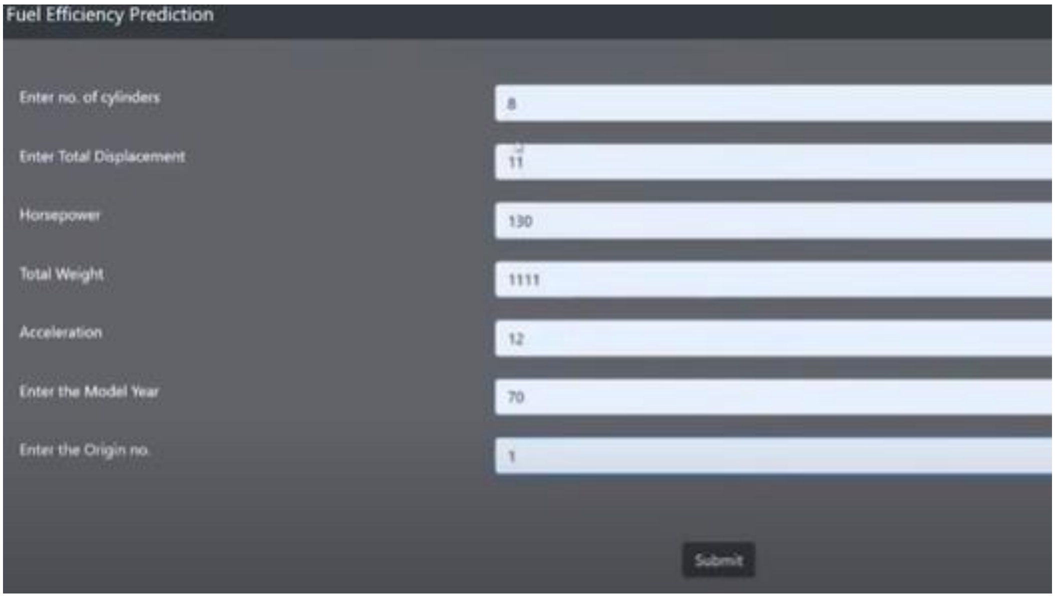


```python
import pickle
from flask import Flask, request, jsonify
from model_files.ml_model import predict_mpg

app = Flask("mpg_prediction")


@app.route('/', methods=['POST'])
def predict():
    vehicle_config = request.get_json()

    with open('./model_files/model.bin', 'rb') as f_in:
        model = pickle.load(f_in)
        f_in.close()
```



```python
predictions = predict_mpg(vehicle_config, model)

response = {
    'mpg_predictions': list(predictions)
}
return jsonify(response)
```

Output:



Fuel Efficiency Prediction

Enter no. of cylinders: 8

Enter Total Displacement: 11

Horsepower: 130

Total Weight: 1111

Acceleration: 12

Enter the Model Year: 70

Enter the Origin no.: 1

Prediction is 41.