Machine Learning Project:- Sales Prediction App

By:- BRIAN ORINA BUNDI

BATCH NUMBER:- 004

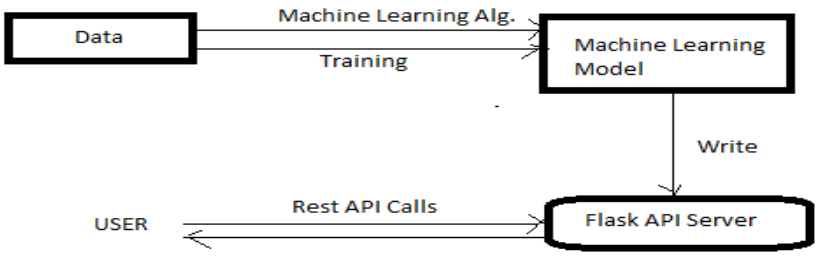
Libraries Used :-

- SkLearn.
- Numpy.
- Pandas.
- Flask .
- Pickle.

Program Implementation overview

This Flow diagram demonstrates the general over view of all the components of the machine learning model and their inter-relation.

- Data – A csv File is used to provide data for the Linear Regression Algorithm used to train the model.
- Machine Learning model- Contains a pickle file that holds the saved model that was trained by the above mentioned data.
- Flask- This is a server that contains an API backend that acts as a middleman between the user requests and the saved model.
- User- People who use the system to get the sales prediction.



A glimpse of the dataset:-

rate	sales_in_first_month	sales_in_second_month	sales_in_third_month
	2	500	300
	4	300	650
four	600	200	400
nine	450	320	650
seven	600	250	350
five	550	200	700

Indepth of each component of the application:-

1. Model.py

This contains code for the machine learning model to predict sales in the third month based on the sales in the first two months.

```
import numpy as np
#import matplotlib.pyplot as plt
import pandas as pd
import pickle

dataset = pd.read_csv('sales.csv')

dataset['rate'].fillna(0, inplace=True)

dataset['sales_in_first_month'].fillna(dataset['sales_in_first_month'].mean(), inplace=True)

X = dataset.iloc[:, :3]

def convert_to_int(word):
    word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6, 'seven':7, 'eight':8,
                 'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0, 0: 0}
    return word_dict[word]

X['rate'] = X['rate'].apply(lambda x : convert_to_int(x))

y = dataset.iloc[:, -1]

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

regressor.fit(X, y)

pickle.dump(regressor, open('model2.pkl','wb'))

model = pickle.load(open('model2.pkl','rb'))
print(model.predict([[4, 300, 500]]))
```

2. HTML/CSS

This contains the HTML template and CSS styling to allow user to enter sales detail and displays the predicted sales in he third month.

```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Sales Prediction Model</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body style="background: #000;">
 <div class="login">
  <h1>Sales Forecasting</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="rate" placeholder="rate" required="required" />
      <input type="text" name="sales in first month" placeholder="sales in first month" required="required" />
      <input type="text" name="sales in second month" placeholder="sales in second month" required="required" />


        <button type="submit" class="btn btn-primary btn-block btn-large">Predict sales in third month</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>


</body>
</html>
```

3. App.py

This contains Flask APIs that receives sales details through GUI or API calls, computes the predicted value based on our model and returns it.

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model2.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

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

    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Sales should be $ {}'.format(output))

@app.route('/results',methods=['POST'])
def results():

    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

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

4. Request.py

This uses requests module to call APIs defined in app.py and displays the returned value.

```
import requests

url = 'http://localhost:5000/results'
r = requests.post(url,json={'rate':5, 'sales_in_first_month':200, 'sales_in_second_month':400})

print(r.json())
```

5. The UI


Sales
Forecasting
5
400
300
Predict sales in third month
Sales should be $ 504.67