Deployment of Linear Regression Model for Predicting Price using Flask

Data used in this Project:

This is a tickets pricing monitoring system. It scrapes tickets pricing data periodically and stores it in a database. Ticket pricing changes based on demand and time, and there can be significant difference in price. We are creating this product mainly with ourselves in mind. Users can set up alarms using an email, choosing an origin and destination (cities), time (date and hour range picker) choosing a price reduction over mean price, etc.
Following is the description for columns in the dataset:

- insert_date: date and time when the price was collected and written in the database
- origin: origin city
- destination: destination city
- start_date: train departure time
- end_date: train arrival time
- train_type: train service name
- price: price
- train_class: ticket class, tourist, business, etc.
- fare: ticket fare, round trip, etc

Project Directory Structure:

The name of the directory used in this project is flask deploy. All the files created one by one and saved into this as;

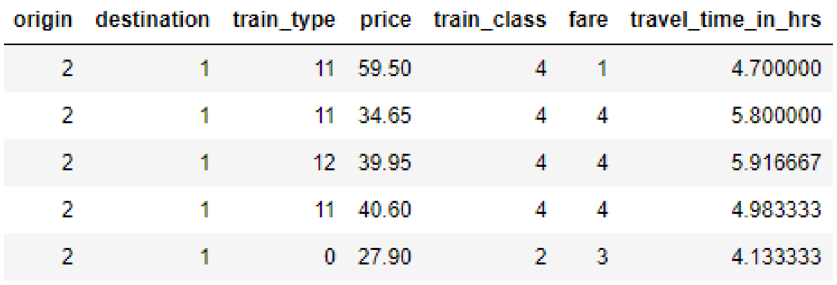
1. model.py-- this file contains the model used to predict the price on the basis of other independent variables.
2. app.py-- this contains the FLASK APIs that receives the price detail through GUI or API calls and compute the predicted value on the basis of model we used.
3. HTML/CSS--this contains the HTML template and CSS styling to allow user to enter sales detail and displays the predicted sales in the third month.

Libraries used in this project:

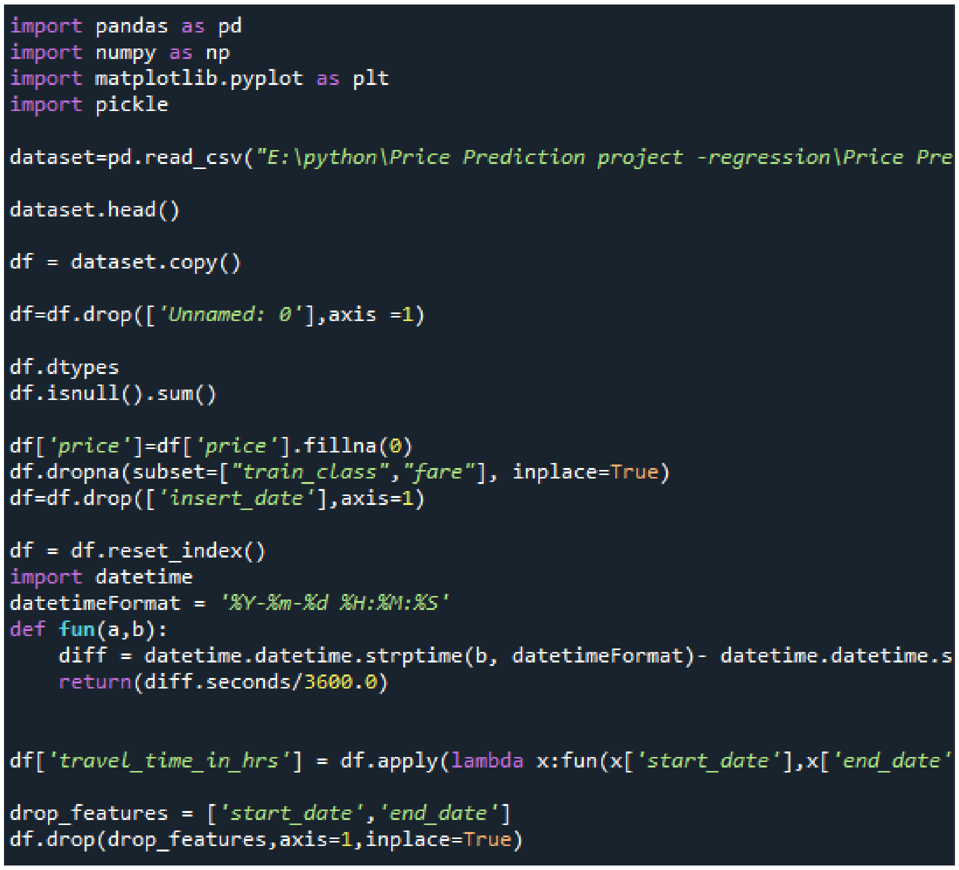
1. scikit-learn
2. pandas
3. numpy
4. flask

Model used in this project & model.py:

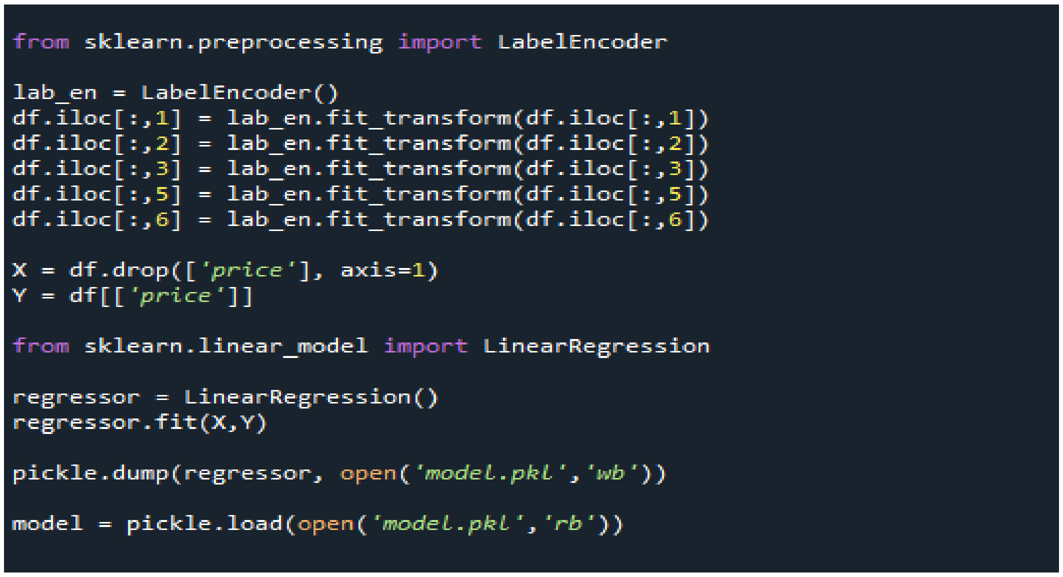
The linear regression modeling technique is used in this project but data is evaluated for missing values, data type and changes are made accordingly. The categorical independent variables are coded, and few variables are dropped from the data before serializing the model. The final used for fitting model looks like this:



origin	destination	train_type	price	train_class	fare	travel_time_in_hrs
2	1	11	59.50	4	1	4.700000
2	1	11	34.65	4	4	5.800000
2	1	12	39.95	4	4	5.916667
2	1	11	40.60	4	4	4.983333
2	1	0	27.90	2	3	4.133333

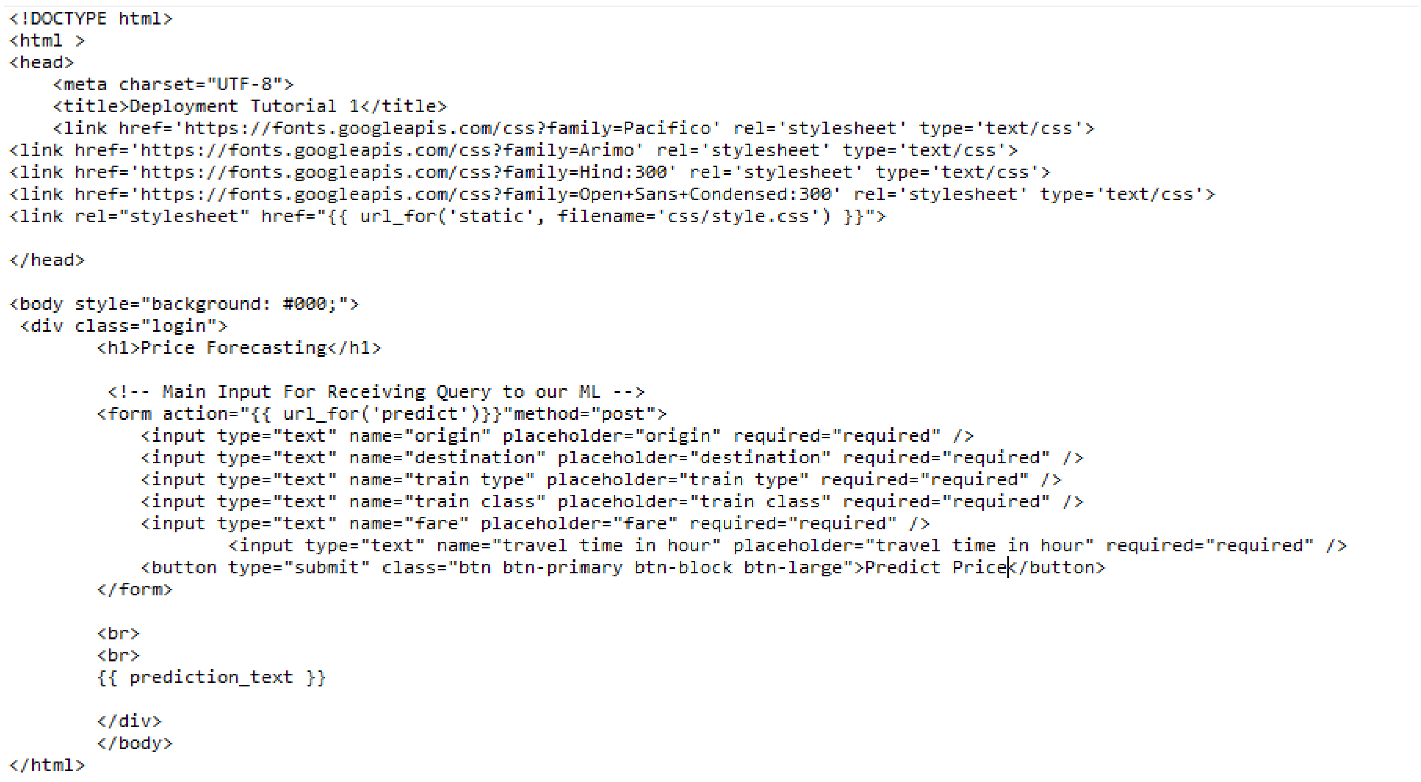


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle

dataset=pd.read_csv("E:\python\Price Prediction project -regression\Price Pre

dataset.head()

df = dataset.copy()

df=df.drop(['Unnamed: 0'],axis =1)

df.dtypes
df.isnull().sum()

df['price']=df['price'].fillna(0)
df.dropna(subset=["train_class","fare"], inplace=True)
df=df.drop(['insert_date'],axis=1)

df = df.reset_index()
import datetime
datetimeFormat = '%Y-%m-%d %H:%M:%S'
def fun(a,b):
    diff = datetime.datetime.strptime(b, datetimeFormat)- datetime.datetime.s
    return(diff.seconds/3600.0)


df['travel_time_in_hrs'] = df.apply(lambda x:fun(x['start_date'],x['end_date'

drop_features = ['start_date','end_date']
df.drop(drop_features,axis=1,inplace=True)
```



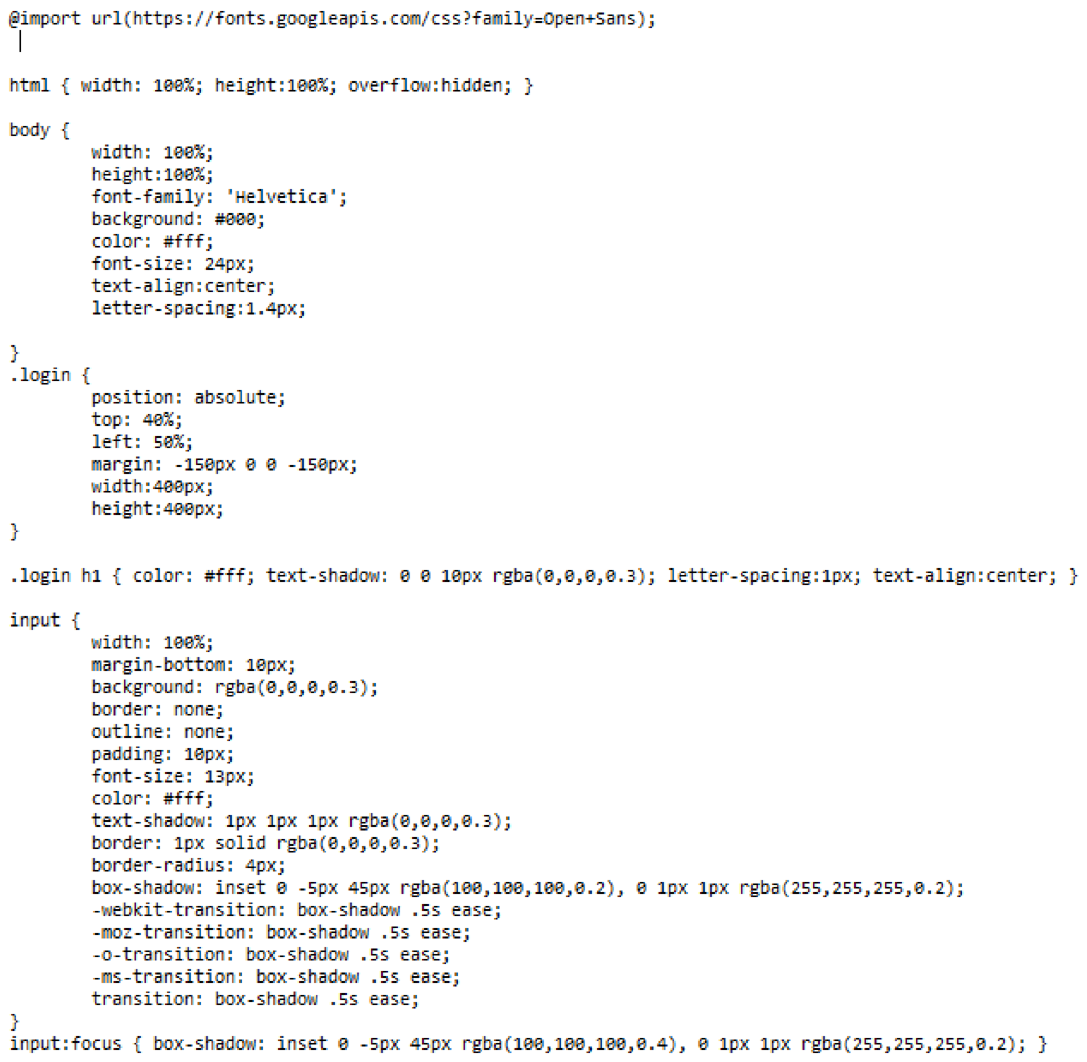
```python
from sklearn.preprocessing import LabelEncoder

lab_en = LabelEncoder()
df.iloc[:,1] = lab_en.fit_transform(df.iloc[:,1])
df.iloc[:,2] = lab_en.fit_transform(df.iloc[:,2])
df.iloc[:,3] = lab_en.fit_transform(df.iloc[:,3])
df.iloc[:,5] = lab_en.fit_transform(df.iloc[:,5])
df.iloc[:,6] = lab_en.fit_transform(df.iloc[:,6])

X = df.drop(['price'], axis=1)
Y = df[['price']]

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X,Y)

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

model = pickle.load(open('model.pkl','rb'))
```

Index.html:



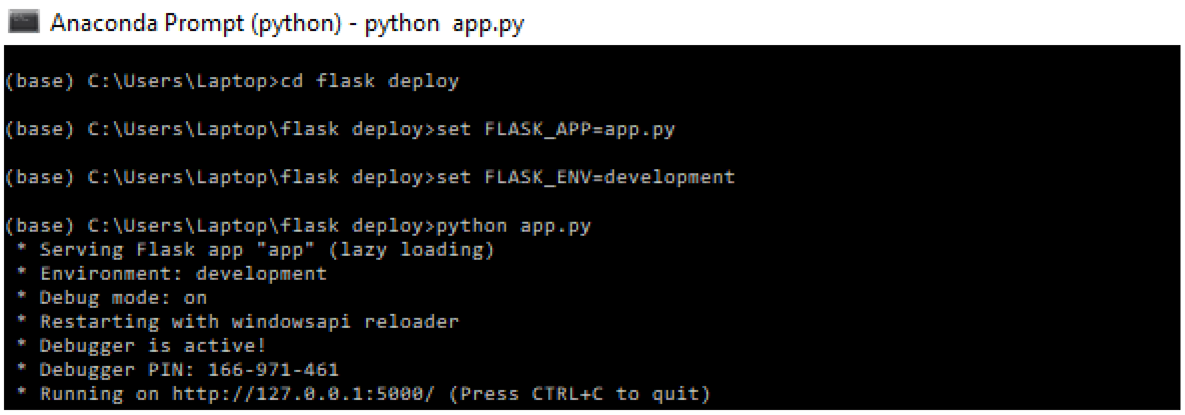
```html
<!DOCTYPE html>
<html >
<head>
    <meta charset="UTF-8">
    <title>Deployment Tutorial 1</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<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>Price Forecasting</h1>

         <!-- Main Input For Receiving Query to our ML -->
        <form action="{{ url_for('predict')}}"method="post">
            <input type="text" name="origin" placeholder="origin" required="required" />
            <input type="text" name="destination" placeholder="destination" required="required" />
            <input type="text" name="train type" placeholder="train type" required="required" />
            <input type="text" name="train class" placeholder="train class" required="required" />
            <input type="text" name="fare" placeholder="fare" required="required" />
                    <input type="text" name="travel time in hour" placeholder="travel time in hour" required="required" />
            <button type="submit" class="btn btn-primary btn-block btn-large">Predict Price</button>
        </form>

        <br>
        <br>
        {{ prediction_text }}

        </div>
        </body>
</html>
```

CSS file:



```css
@import url(https://fonts.googleapis.com/css?family=Open+Sans);

html { width: 100%; height:100%; overflow:hidden; }

body {
        width: 100%;
        height:100%;
        font-family: 'Helvetica';
        background: #000;
        color: #fff;
        font-size: 24px;
        text-align:center;
        letter-spacing:1.4px;

}
.login {
        position: absolute;
        top: 40%;
        left: 50%;
        margin: -150px 0 0 -150px;
        width:400px;
        height:400px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px; text-align:center; }

input {
        width: 100%;
        margin-bottom: 10px;
        background: rgba(0,0,0,0.3);
        border: none;
        outline: none;
        padding: 10px;
        font-size: 13px;
        color: #fff;
        text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
        border: 1px solid rgba(0,0,0,0.3);
        border-radius: 4px;
        box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
        -webkit-transition: box-shadow .5s ease;
        -moz-transition: box-shadow .5s ease;
        -o-transition: box-shadow .5s ease;
        -ms-transition: box-shadow .5s ease;
        transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(255,255,255,0.2); }
```

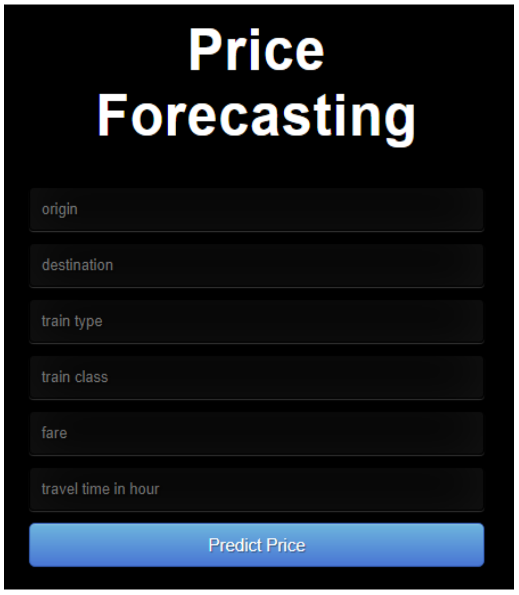
Deployment using Anaconda Prompt:



Anaconda Prompt (python) - python app.py

```
(base) C:\Users\Laptop>cd flask deploy

(base) C:\Users\Laptop\flask deploy>set FLASK_APP=app.py

(base) C:\Users\Laptop\flask deploy>set FLASK_ENV=development

(base) C:\Users\Laptop\flask deploy>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 166-971-461
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Result:



Price Forecasting

origin

destination

train type

train class

fare

travel time in hour