DYNAMIC PRICING PREDICTION FOR CABS USING IBM WATSON

TEAM MEMBERS

- THRIVENI PEGYAPURAM
- NAGAVELLI MANISH KUMAR
- N. ADITHYA VARDHAN

INTRODUCTION:

Many organizations do not have a direct role in travel and tourism but offer related products and services. Some examples would be offering travel insurance, parking facilities at airports, theatre and event tickets, car hire, and travel by rail or coach to airports, etc. at competitive rates. There are various different forms of dynamic pricing:

1. Peak Pricing – This is a strategy that is common in transportation businesses. Airlines are a good example. Airlines often charge a higher price to travel during rush hour mostly on weekdays and sometimes on weekends.

2. Surge Pricing – Companies such as Uber respond dynamically to changes in supply and demand in order to price their services differently. Like most of us have noticed, this frequently happens on stormy evenings and nights when more people request for cabs. Taxify also not so long ago introduced dynamic pricing to ensure the drivers are encouraged to go online and offer services when the demand is high.

LITERATURE SURVEY:

In this chapter, we discuss various applications and methods which inspired us to build our

project. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of information like the technological stack, algorithms, and shortcomings of our project which led us to build a better project Predicting price of a used cars has been studied extensively in various researches discussed, in her paper written for Master thesis [2], that regression model that was built using Random Forest Regression can predict the price of a car that has been leased with better precision than multivariate regression or some simple multiple regression. This is on the grounds that Random Forest Regression is better in dealing with datasets with more dimensions and it is less prone to overfitting and underfitting. The weakness of this research is that a change of simple regression with more advanced Random Forest regression was not shown in basic indicators like mean, variance or standard deviation.

HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware Requirements:

- Processor: Minimum 1 GHz; Recommended 2GHz or more.
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 32 GB; Recommended 64 GB or more.
- Memory (RAM): Minimum 1 GB; Recommended 4 GB or above.

Software Requirements:

1. Anaconda Navigator
2. Spyder
3. Jupiter Notebook

OBJECTIVES:

Write what are all the technical aspects that students would get if they complete this project.

- Knowledge on Machine Learning Algorithms.
- Knowledge on Python Language with Machine Learning
- Knowledge on Statistics and Graphs and their relations
- You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process / clean the data using different data pre-processing techniques.
- Applying different algorithms according to the dataset and based on visualization.
- Real Time Analysis of Project
- Building an ease of User Interface (UI)
- Navigation of ideas towards other projects(creativeness)
- Knowledge on building ML Model.
- You will be able to know how to find the accuracy of the model.
- How to Build web applications using the Flask framework.
- To find correlations between each variable and the target variable.
- To build a linear regression model.
- To calculate efficiency of the model and errors in predictions

EXPERIMENTAL INVESTIGATIONS:

```
#Importing Libraries

import numpy as np #linear algebra
import pandas as pd #data processing
import matplotlib.pyplot as plt
import seaborn as sns #used for data visualization
import pickle
from collections import Counter as c # return counts
from sklearn.preprocessing import LabelEncoder #importing the LabelEncoding from sklearn
from sklearn.model_selection import train_test_split #split data in train and test array
from sklearn.ensemble import RandomForestRegressor #regression ML algorithm
```

```
#Cab rides represent the for the type of cab,location,name etc.
rides_df.head()
```

	distance	cab_type	time_stamp	destination	source	price	surge_multiplier	id	product_id	name
0	0.44	Lyft	1544952607890	North Station	Haymarket Square	5.0	1.0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	lyft_line	Shared
1	0.44	Lyft	1543284023677	North Station	Haymarket Square	11.0	1.0	4bd23055-6827-41c6-b23b-3c491f24e74d	lyft_premier	Lux
2	0.44	Lyft	1543366822198	North Station	Haymarket Square	7.0	1.0	981a3613-77af-4620-a42a-0c0866077d1e	lyft	Lyft
3	0.44	Lyft	1543553582749	North Station	Haymarket Square	26.0	1.0	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	lyft_luxsuv	Lux Black XL
4	0.44	Lyft	1543463360223	North Station	Haymarket Square	9.0	1.0	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	lyft_plus	Lyft XL

```
#Weather data represents the temperature ,humidity etc.
weather_df.head()
```

	temp	location	clouds	pressure	rain	time_stamp	humidity	wind
0	42.42	Back Bay	1.0	1012.14	0.1228	1545003901	0.77	11.25
1	42.43	Beacon Hill	1.0	1012.15	0.1846	1545003901	0.76	11.32
2	42.50	Boston University	1.0	1012.15	0.1089	1545003901	0.76	11.07
3	42.11	Fenway	1.0	1012.13	0.0969	1545003901	0.77	11.09
4	43.13	Financial District	1.0	1012.14	0.1786	1545003901	0.75	11.49

Label Encoding

In [57]:
```
data1=data.copy()
from sklearn.preprocessing import LabelEncoder #importing the LabelEncoding from sklearn
x='*'
for i in cat:#Looping through all the categorical columns
    print("LABEL ENCODING OF:",i)
    LE = LabelEncoder()#creating an object of LabelEncoder
    print(c(data[i])) #getting the classes values before transformation
    data[i] = LE.fit_transform(data[i]) # trannsforming our text classes to numerical values
    print(c(data[i])) #getting the classes values after transformation
    print(x*100)
```

```
LABEL ENCODING OF: cab_type
Counter({'Uber': 330568, 'Lyft': 307408})
Counter({1: 330568, 0: 307408})
****************************************************************************************************
LABEL ENCODING OF: destination
Counter({'Financial District': 54192, 'Back Bay': 53190, 'Theatre District': 53189, 'Boston University': 53171, 'Haymarket Square': 53171, 'Fenway': 53166, 'Northeastern
University': 53165, 'North End': 53164, 'South Station': 53159, 'West End': 52992, 'Beacon Hill': 52840, 'North Station': 52577})
Counter({4: 54192, 0: 53190, 10: 53189, 2: 53171, 5: 53171, 3: 53166, 8: 53165, 6: 53164, 9: 53159, 11: 52992, 1: 52840, 7: 52577})
****************************************************************************************************
LABEL ENCODING OF: source
Counter({'Financial District': 54197, 'Back Bay': 53201, 'Theatre District': 53201, 'Boston University': 53172, 'North End': 53171, 'Fenway': 53166, 'Northeastern Univers
ity': 53164, 'South Station': 53160, 'Haymarket Square': 53147, 'West End': 52980, 'Beacon Hill': 52841, 'North Station': 52576})
Counter({4: 54197, 0: 53201, 10: 53201, 2: 53172, 6: 53171, 3: 53166, 8: 53164, 9: 53160, 5: 53147, 11: 52980, 1: 52841, 7: 52576})
****************************************************************************************************
LABEL ENCODING OF: id
Counter({'424553bb-7174-41ea-aeb4-fe06d4f4b9d7': 1, '4bd23055-6827-41c6-b23b-3c491f24e74d': 1, '981a3613-77af-4620-a42a-0c0866077d1e': 1, 'c2d88af2-d278-4bfd-a8d0-29ca77c
```

Splitting dataset into train and test

In [63]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
print(x_train.shape)
print(x_test.shape)
```

```
(510380, 5)
(127596, 5)
```

In [64]:
```
from sklearn.ensemble import RandomForestRegressor
rand=RandomForestRegressor(n_estimators=20,random_state=52,n_jobs=-1,max_depth=4)
rand.fit(x_train,y_train)
```

```
<ipython-input-64-57509395cdf7>:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for examp
le using ravel().
  rand.fit(x_train,y_train)
```

Out[64]:
```
RandomForestRegressor(max_depth=4, n_estimators=20, n_jobs=-1, random_state=52)
```

In [80]:
```
x_train.shape
```

Out[80]:
```
(510380, 5)
```

Predicting the result

In [66]:
```
ypred=rand.predict(x_test)
print(ypred)
```

```
[33.44544798 19.16381383  9.54753035 ...  6.02421004 26.79738243
 17.55244465]
```

Score of the model

In [67]:
```
rand.score(x_train,y_train)
```

Out[67]:
```
0.7575275520145969
```

Saving our model

In [68]:
```
import pickle
pickle.dump(rand, open("model.pkl", "wb"))
```

In [69]:
```
!pip install ibm_watson_machine_learning
```

```
Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (1.0.166)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (2021.10.8)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (1.26.6)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (2.25.1)
Requirement already satisfied: pandas<1.3.0,>=0.24.2 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (1.2.4)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (20.9)
Requirement already satisfied: ibm-cos-sdk==2.7.* in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (2.7.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-cos-sdk==2.7.*->ibm_watson_machine_learning)
(0.10.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.7.0 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-cos-sdk==2.7.*->ibm_watson_machine_lea
rning) (2.7.0)
Requirement already satisfied: ibm-cos-sdk-core==2.7.0 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-cos-sdk==2.7.*->ibm_watson_machine_learning)
(2.7.0)
Requirement already satisfied: docutils<0.16,>=0.10 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-cos-sdk-core==2.7.0->ibm-cos-sdk==2.7.*->ibm_wa
tson_machine_learning) (0.15.2)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-cos-sdk-core==2.7.0->ibm-cos-sdk==2.7.*-
>ibm_watson_machine_learning) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from pandas<1.3.0,>=0.24.2->ibm_watson_machine_learning) (2021.
1)
```

In [70]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
                   "url": "https://us-south.ml.cloud.ibm.com",
                   "apikey":"l0ArGgOyZ3AWyBtW5SQG-9PPK8SgWoamljoLcqyw3CaR"
                  }
client = APIClient(wml_credentials)
```

In [71]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    #print(space)
    return(next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [72]:
```
space_uid = guid_from_space_name(client,'models')
print("Space UID=" + space_uid)
```

```
Space UID=b6db309d-e71f-4ec5-a984-6e5086007a83
```

In [73]:
```
client.set.default_space(space_uid)
```

Out[73]: 'SUCCESS'

In [74]:
```
client.software_specifications.list()
```

```
------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
```

In [75]:
```
software_spec_uid = client.software_specifications.get_uid_by_name("default_py3.8")
software_spec_uid
```

Out[75]: 'ab9e1b80-f2ce-592c-a7d2-4f2344f77194'

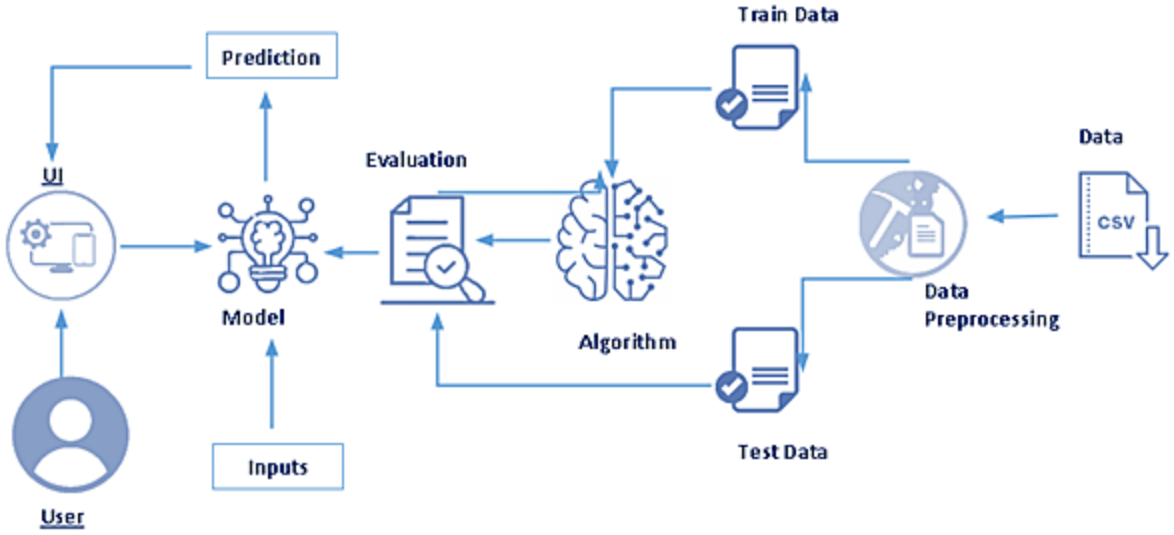
In [76]:
```
model_details = client.repository.store_model(model=rand,meta_props={
client.repository.ModelMetaNames.NAME:"cab_rides",
client.repository.ModelMetaNames.TYPE:"scikit-learn_0.23",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid }

                                              )
model_id =client.repository.get_model_uid(model_details)
```

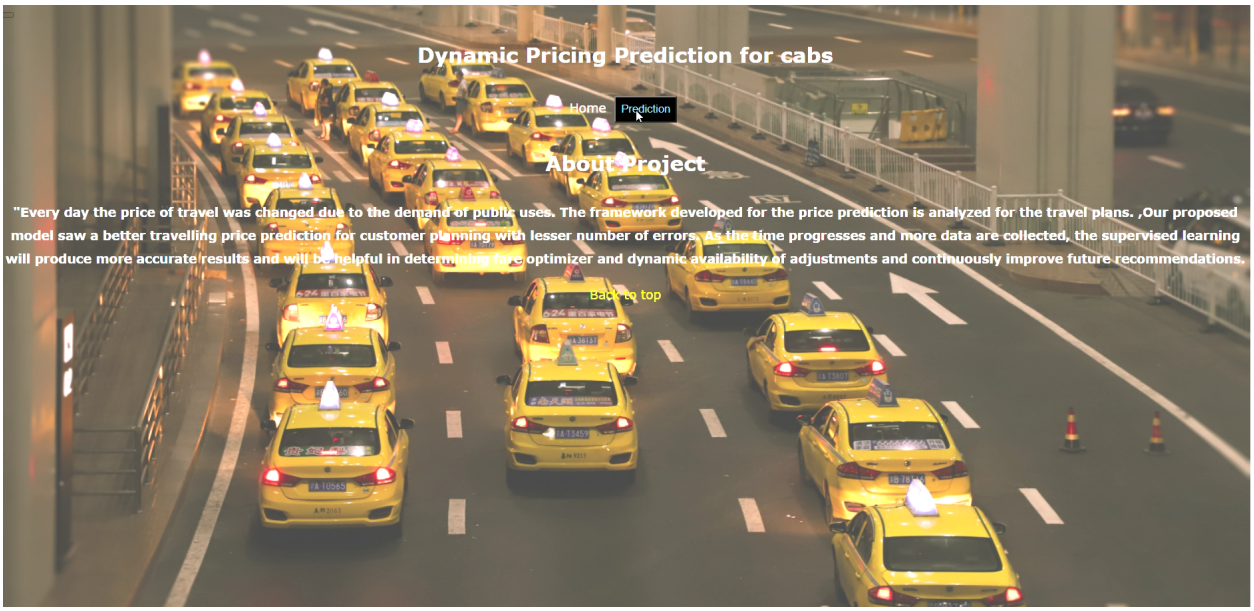
In [77]:
```
model_id
```

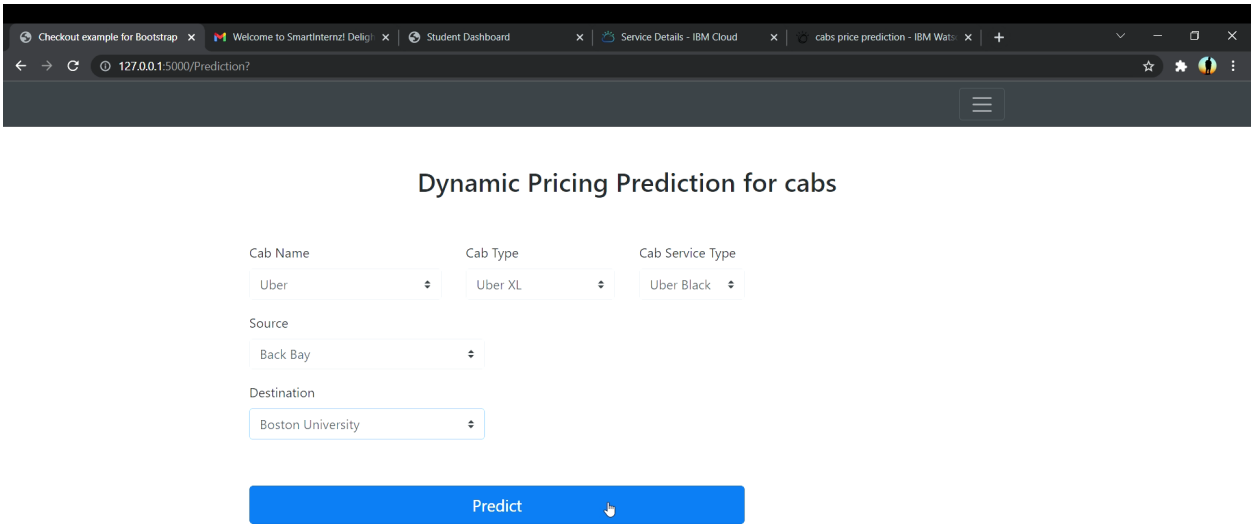
Out[77]: '5eb1702c-7cb6-4441-a889-45ab70a54822'

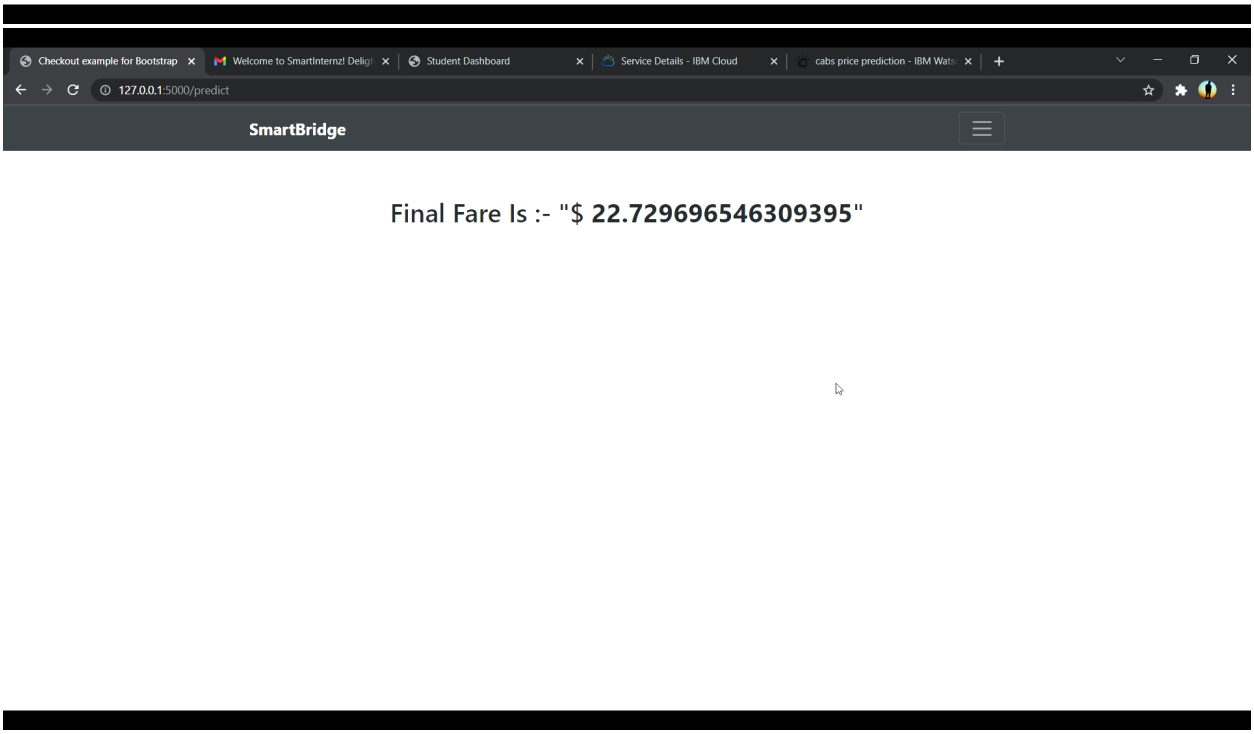
ARCHITECTURE:



RESULT




Checkout example for Bootstrap
Welcome to SmartInternz! Deligh
Student Dashboard
Service Details - IBM Cloud
cabs price prediction - IBM Wats
127.0.0.1:5000/Prediction?
Dynamic Pricing Prediction for cabs
Cab Name
Uber
Cab Type
Uber XL
Cab Service Type
Uber Black
Source
Back Bay
Destination
Boston University
Predict


Checkout example for Bootstrap
Welcome to SmartInternz! Deligh
Student Dashboard
Service Details - IBM Cloud
cabs price prediction - IBM Wats
127.0.0.1:5000/predict
SmartBridge
Final Fare Is :- "$ 22.729696546309395"

ADVANTAGES & DISADVANTAGES:

Advantages:

1. They need to work more on their network.
2. There is a lack of communications with the drivers.
3. Cabs brings a whole lot of exclusive perks for corporate employees.
4. Insurance
5. Your safety
6. Pricing automation.
7. Increased competitiveness

Disadvantages:

1. There are payment issues in this cab service.
2. You have a big responsibility for your passengers.
3. Cab drivers don’t make much money.
4. You don’t have regular schedule.
5. Cab drivers have to work at nighttime
6. Some cab drivers even have to work on weekends and holidays.

7. Can be problematic for your family life.
8. Drivers sometimes try to game the system for bigger profits
9. Customer alientation and backlash

CONCLUSION:

Every day the price of travel was changed due to the demand for public uses. The framework developed for the price prediction is analyzed for the travel plans. For the same travel plan offered at a fixed price for a particular group of customers, our proposed model saw a final fare with a lesser number of errors in predicting customer planning. As time progresses and more data are collected, the supervised learning will produce more accurate results and will be helpful in determining fare optimizer and dynamic availability of adjustments and continuously improve future recommendations.

FUTURE SCOPE:

- The dependent algorithm variable turned out to be 'ride fare,' while 'travel distance' and 'travel time' are the independent variables.
- The random forest model for prediction of dynamic price of trips is providing an efficiency of 93.40%.
- It is better suited for the prediction of target variable which is trip fare, and it performs very well.
- Further this work can be carried out using different machine learning algorithms and

techniques in order to get higher efficiency and lower errors.

BIBILOGRAPHY:

https://www.youtube.com/watch?v=5mDYijMfSzs