DYNAMIC PRICING

Team Members:

- AbdElrahman Magdy (Team Leader)
- AbdElrahman Salah
- Doha Attio
- Enas Abdelfattah
- Noha Nagy El-Barkouky

Contents

Dynamic Pricing

Definition:

An e-commerce and retail strategy applies variable pricing instead of the typical fixed pricing. As more data analyzed, optimal prices for items are calculated. The time between price changes depends upon the business and item, but can be as often as every day, or even every hour.

This strategy is not a new thing. In the past, pricing was influenced solely by demand and supply in a locality. How many people want to buy the product? How much inventory of the product is currently available? Is the item perishable? Will the item be replaced by a newer version at any point in the near future? These are all the types of questions that play a big role in traditional pricing.

It uses advanced data, including data from some of the previously posed questions. Dynamic pricing is often an automated process that looks at more than just the traditional factors.

Why is Dynamic Pricing Important in E-Commerce?

It has become critical in e-commerce, mostly due to automation. Where as in a store, employees would have to physically change the pricing on thousands of items (as well as create new pricing display information); Online the price of an item can be dynamically adjusted without much cost to the business.

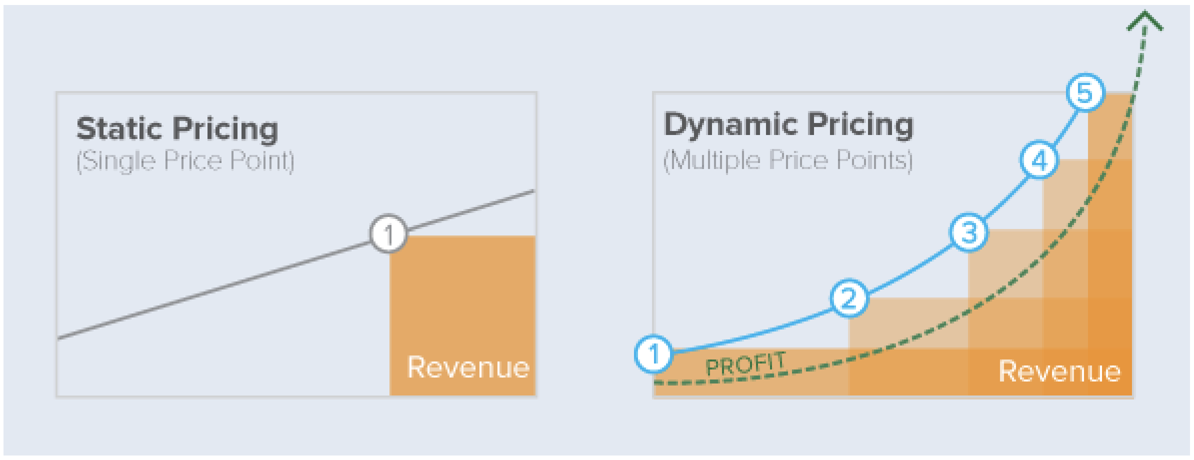


Figure 1: Dynamic pricing Vs. Static pricing

Let us take a look at some of the benefits that dynamic pricing provides to e-commerce businesses:

1. <u>It Gives You Greater Control on Your Pricing Strategy</u>

   A common argument against dynamic pricing is that it reduces your control over product pricing. In reality, it has an opposite effect. As a retailer using dynamic pricing, you will have access to real-time price trends across thousands of products in your industry. You will be able to see the pricing changes of your competitors, and will have a clear idea of the supply and demand of individual products. This will help you set the right prices for different products and maximize your revenues.

2. It Allows Flexibility Without Compromising Your Brand Value

Brands can protect, and even strengthen, their brand value by implementing dynamic pricing. Retailers can set a price floor that reflects their brand value and allows them the flexibility to stay profitable. They can use dynamic pricing to launch seasonal and promotional offers as well, while still remaining profitable (something quite difficult to achieve with a flat pricing model).

3. It Saves You Money in the Long Run

Dynamic pricing is based on the changes in real-time product supply and demand. It takes into account the price fluctuations in the market, monitors competitor activity and individual product demand and supply. As a result, it gives ecommerce retailers the right data and information that can be used to set optimal product prices and stay profitable despite the price fluctuations.

This saves retailers' money in the long run. Since all the calculations are done by web based software and applications, there is no need for spending money in manual calculations and administrative activities. The reduction in these overheads also adds to your profitability in the long run.

For example, Walmart uses dynamic pricing and changes its product prices almost 50,000 times a month. Using this pricing model, its global sales grew by 30% in 2013 and the trend continued in 2014 as well. Amazon also saw a 27.2% increase in revenues and ended up as one of the top 10 retailers in the US for the first time.

4. It Can Be Managed Effectively with the Right Software

Monitoring hundreds of thousands of products and keeping an eye on the real-time supply and demand trends is a highly complex and challenging task, beyond the scope of most ecommerce businesses. However, it can be easily managed with the right software. Wiser, for example, is a web-based application designed to monitor, calculate and manage dynamic pricing models based on real-time supply and demand trends.

It takes the guesswork out of dynamic pricing and automates the whole process to provide you accurate data that can be used to set optimal product prices. Wiser has also helped global corporations like 3M, McAfee and at&t with dynamic pricing and MAP enforcement over the last two years.

5. It's Not Error Free, But You're Still in Control

Dynamic pricing is based on supply and demand changes. But like any other technology based forecast, there is potential for error in dynamic pricing algorithms as well. However, even if the proposed pricing is inaccurate, it's still just a proposal. You remain in control all the time and can review the pricing changes that your application recommends.

Moreover, the experiences of companies like Amazon, Best Buy and Walmart indicate that potential errors are not only easily manageable but also usually do not have a significant impact on the overall profits, since the changes are so frequent.

ecommerce retailers have grown in numbers over the last few years. With increased competition, they face the tough challenge of maximizing profits while keeping their prices competitive. Dynamic pricing is the ideal solution to this problem since it takes into account the changes in supply and demand, and recommends the optimal pricing structure. If implemented for a sustained period, this pricing strategy can significantly boost your overall revenues and profitability.

Common Dynamic Pricing Strategies:

Automated dynamic pricing can consider many data when setting prices. In fact, the largest retailers use a combination of data in their algorithms to determine the final price of an item. Some of the different data types that can be used in dynamic pricing strategies include:

- Supply and demand based on locality

  As an e-commerce business, the supply and demand of different items may differ based on geography. Looking at these factors, prices can be adopted by country, state, or city.

- Time-based dynamic pricing

  Time can also play a role in dynamic pricing. Consider how bars and pubs offer a happy hour, during which the prices on drinks and food are discounted.

- Competitor pricing

  What are the prices being offered by competitors for the same product? Businesses could ensure that their pricing is always lower, or even raise their price to match inflated competitor pricing.

- Customer behavior

  The way that a customer behaves can also influence pricing. If a customer has come to your website three times and looked at the same product, perhaps a small discount would convince them to buy. Using customer actions to trigger pricing changes can be an excellent way to run tests.

- Segmented Dynamic Pricing

  Customer data can also be used to influence dynamic prices. If a customer has filled out a survey and indicated that they have high levels of income, you can use that information to charge a higher price.

- Peak pricing

  Peak pricing refers to changes in price based on current supply. If your company finds that they are unable to fulfill all requests for a specific product, increasing the price of the item makes sense. For example, Uber uses peak dynamic pricing to increase prices when they are at heavy load, and drivers have too many ride requests to adequately serve each person.

  Dynamic pricing allows for a lot of creativity. While automation and machine learning ultimately drive measurable growth, there is an almost unending number of ways that data can be used to influence price.

Business Understanding

The data set includes complete taxicab trip records for calendar year 2017. Each record includes information about the driver, vehicle, time and distance traveled, trip origin location, trip destination location, and the fare.

The objective here is to use the concept of dynamic pricing in order to get the more accurate price for each trip considering many features like distance, vehicle, time and so on.

Challenges & Assumptions

- Data size.
- Classes boundaries.
- Clustering features.
- Dynamic Pricing Regression Model.
- Generating more features.

Business Goals

- The service will offer dynamic but upfront fares to passengers.
- Increasing fares during peak hours, at some high-traffic locations such as the airport and integrated resorts, and for commuters who are willing to book an immediate ride. This will attract more drivers and nudge some commuters to explore alternative modes of transport.
- Ability to offer suitable discounts at the off peak time, and to increase the rate in the peak time.

Data Understanding

The data set is 4GB in size. File format is in CSV. The data contains 32 features. Each record includes information about the driver, vehicle, time and distance traveled, trip origin location, trip destination location, and the fare. The data is publicly available on Kaggle (1).

As part of the agency's commitment to encourage innovation in the for-hire vehicle industry, DFHV seeks to introduce a dynamic pricing system for taxicabs. DFHV wants to implement three pricing tiers (eg, off-peak-normal-peak, or low medium high) based on trip data. The tiers could be based on times or areas that have more or less demand for taxicab service. As the taxicab regulator, DFHV sets taxicab rates and would establish parameters for dynamic pricing, which taxicabs would follow.

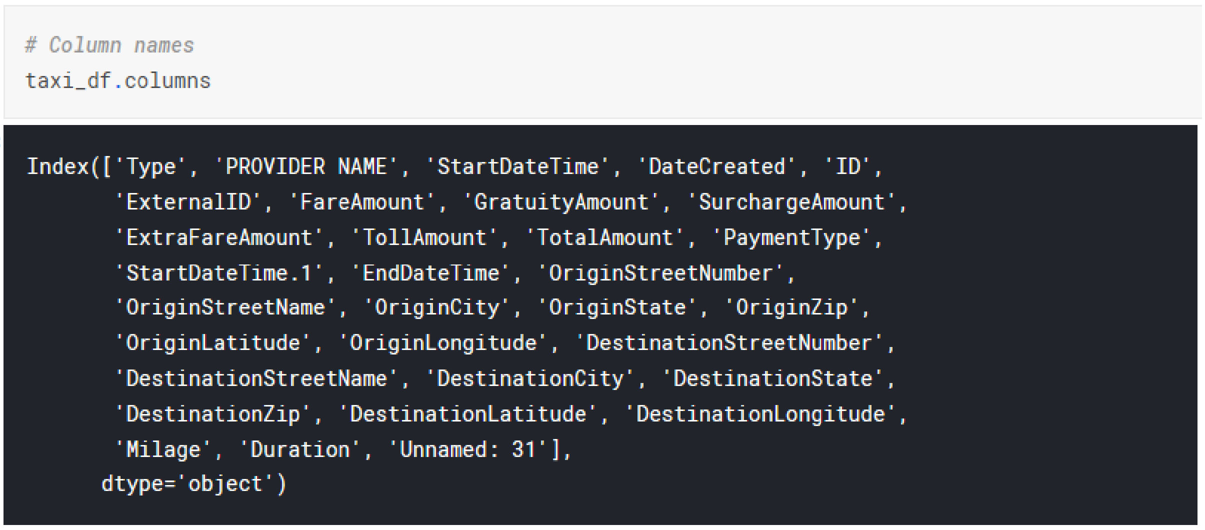
Notebook on Kaggle

To label each hour in a day with off-peak-normal-peak, they applied K-Means clustering in python with scikit-learn. In the process of cleaning data, they removed the null and wrong data from dataset. Due to the different pattern between weekdays and weekends, they divided total dataset into two parts and develop clustering models separately. As a result, we found that peak time is 12-19 am and the off-peak time is 0-7 am on weekdays. For weekend, the peak time is 0-2 am and 14-19 am, and the off-peak time is 4-9 am.

1) https://www.kaggle.com/bvc5283/data-challenge/data

Data Description

The data set is csv file comma delimited, includes 32 columns descripting the trip attributes for example the distance, start date time, end date time, origin city, destination city, address, provider name and so on, and we have 11937645 record.



```
# Column names
taxi_df.columns
```

```
Index(['Type', 'PROVIDER NAME', 'StartDateTime', 'DateCreated', 'ID',
       'ExternalID', 'FareAmount', 'GratuityAmount', 'SurchargeAmount',
       'ExtraFareAmount', 'TollAmount', 'TotalAmount', 'PaymentType',
       'StartDateTime.1', 'EndDateTime', 'OriginStreetNumber',
       'OriginStreetName', 'OriginCity', 'OriginState', 'OriginZip',
       'OriginLatitude', 'OriginLongitude', 'DestinationStreetNumber',
       'DestinationStreetName', 'DestinationCity', 'DestinationState',
       'DestinationZip', 'DestinationLatitude', 'DestinationLongitude',
       'Milage', 'Duration', 'Unnamed: 31'],
      dtype='object')
```

Figure 2: Dataset features

Below description for the most important columns:

Column Name	Description
PROVIDER NAME	The taxi company provider
StartDateTime	The date & time when the trip started
DateCreated	Date when the trip was created
TotalAmount	Trip fees
EndDateTime	The date & time when the trip ended
Milage	Distance by miles
Duration	Estimated time for the ride
OriginStreetNumber	Street number for source address
OriginStreetName	Street Name for source address
OriginCity	City Name for source address
OriginState	State name for source address
OriginZip	Zip code for source address
OriginLatitude	Source latitude coordinates
OriginLongitude	Source longitude coordinates
DestinationStreetNumber	Street number for destination address
DestinationStreetName	Street Name for destination address
DestinationCity	City Name for destination address
DestinationState	State name for destination address
DestinationZip	Zip code for destination address
DestinationLatitude	destination latitude coordinates
DestinationLongitude	destination longitude coordinates

Table 1: dataset features description

Data Exploration

We got some insights from the data set to understand it well.

- The highest sum('FareAmount') is vendor 'UVC' and the lowest one is 'My taxi control'.
- This should reflect on the number of rides and the offers that customer needs from the vendor, also the covered cities/ areas.

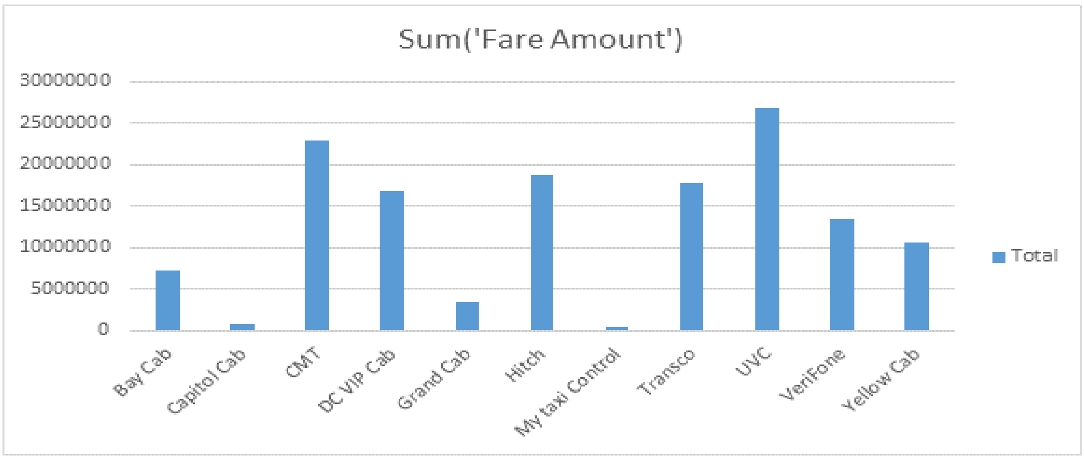


Figure 3: Chart shows number fairs per each provider

Heat map for number of rides across the week days (Wednesday – Friday) and across the whole day (0 -- 23), From below figure we can see,

1. The highest number of rides is at the working days (Monday -- Friday) and especially afternoon.
2. In all days, there is an off peak from 3-6 AM, the lowest number of rides here.
3. In weekend days, off peak and peak boarders are very different from the working days.
4. In addition, the max/ min number of rides in peak/ off peak in weekend and working days are very different; the max number of rides at the weekend is apparently smaller than the max number of rides at the working days.

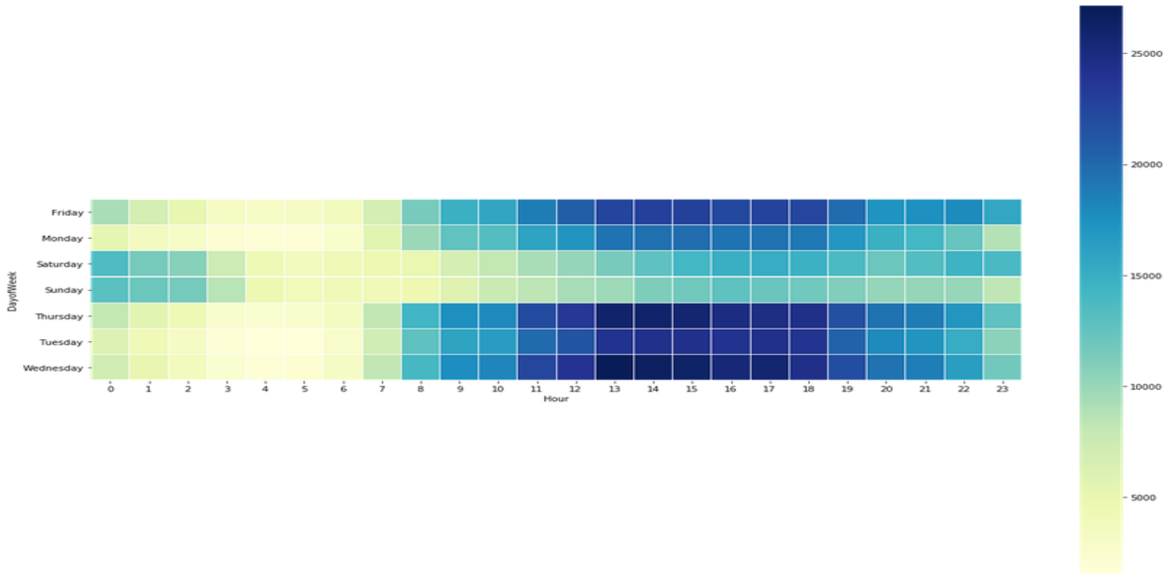


Figure 4: rides traffic according to weekdays and day hours

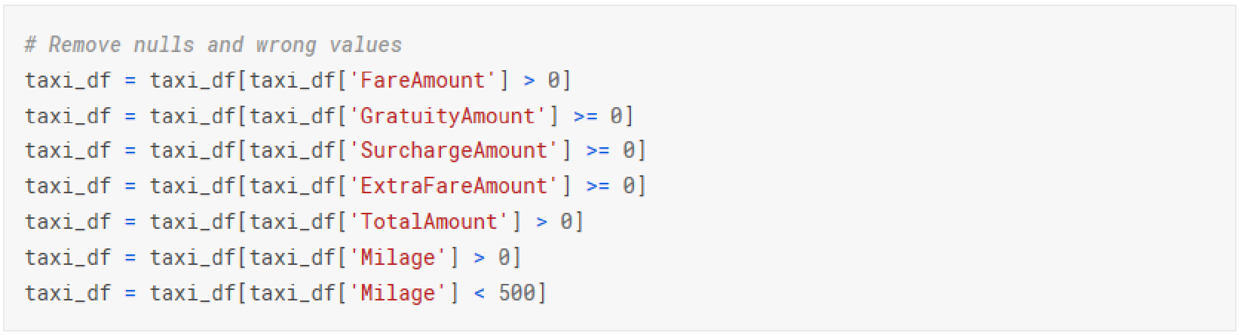
Data Preparation

Data Selection

First, we removed the useless columns 'DateCreated', 'StartDateTime.1' and 'Unnamed: 31'. The dataset became 29 columns only. The features highly contributed in estimating fair total amount are Mileage, Duration, Origin latitude, Origin Longitude, date and time for the trip. Start date will not be used as is, we extracted another three features from day hour, weekday and is this weekday weekend or not.

Data Cleaning

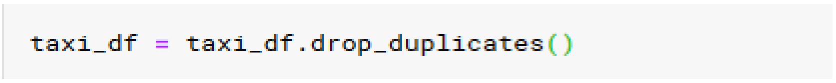
Then we removed all wrong values as shown figure below .



```
# Remove nulls and wrong values
taxi_df = taxi_df[taxi_df['FareAmount'] > 0]
taxi_df = taxi_df[taxi_df['GratuityAmount'] >= 0]
taxi_df = taxi_df[taxi_df['SurchargeAmount'] >= 0]
taxi_df = taxi_df[taxi_df['ExtraFareAmount'] >= 0]
taxi_df = taxi_df[taxi_df['TotalAmount'] > 0]
taxi_df = taxi_df[taxi_df['Milage'] > 0]
taxi_df = taxi_df[taxi_df['Milage'] < 500]
```

Figure 5: Removing rows with zero value

Then we removed the duplicated record.



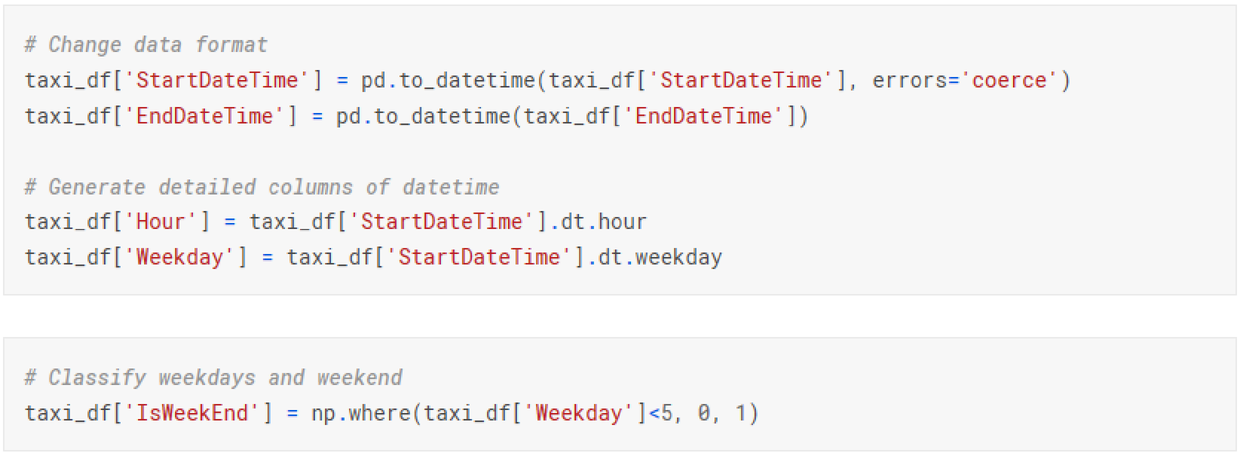
```
taxi_df = taxi_df.drop_duplicates()
```

Figure 6: Removing duplicates rows

After applying previous cleaning steps, the total number of samples is 10957863 sample.

Features Engineering

We generated new three features ‘Hour’, ‘Weekday’ and IsWeekday from ‘StartDateTime’ an new column ‘IsWeekEnd’ t identify the day is weekend or not, then we removed the wrong values.

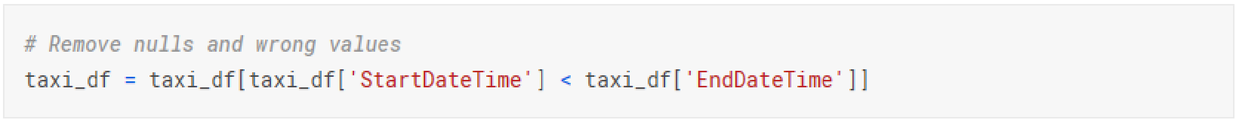


```
# Change data format
taxi_df['StartDateTime'] = pd.to_datetime(taxi_df['StartDateTime'], errors='coerce')
taxi_df['EndDateTime'] = pd.to_datetime(taxi_df['EndDateTime'])

# Generate detailed columns of datetime
taxi_df['Hour'] = taxi_df['StartDateTime'].dt.hour
taxi_df['Weekday'] = taxi_df['StartDateTime'].dt.weekday
```

```
# Classify weekdays and weekend
taxi_df['IsWeekEnd'] = np.where(taxi_df['Weekday']<5, 0, 1)
```

Figure 7: creating Hour and Weekday from start date feature



```
# Remove nulls and wrong values
taxi_df = taxi_df[taxi_df['StartDateTime'] < taxi_df['EndDateTime']]
```

Figure 8: removing records where end date greater than start date

The date set became 10488990 record.

Modeling

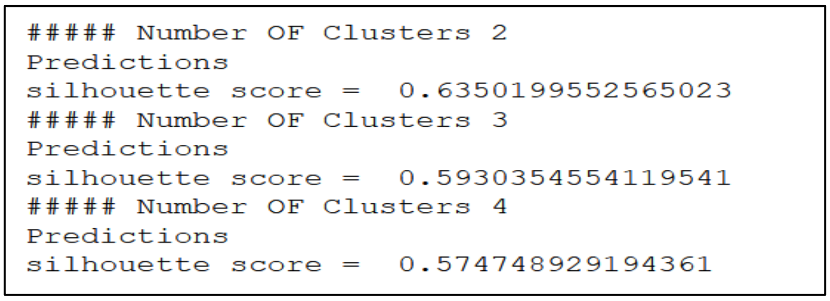
Modeling Techniques Selection

We implemented two different approaches for dynamic pricing model,

- <u>Clustering & regression Model</u>

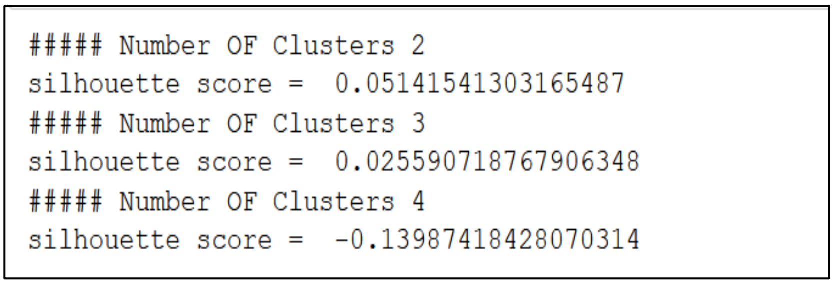
  We applied clustering on the date then we applied the regression model on each cluster to get the predicted total amount for the trip. We thought that clustering before regression will increase the accuracy of predication. As efficient dynamic pricing depends on splitting customers into segments and tread them based in that.

  We tried two clustering algorithms K-Means and GMM (Gaussian Mixture Model). We form a new data frame with features ('IsWeekday', 'Hour', 'Weekday', 'Milage' and ‘Duration’). We run each algorithm on different Ks range from two to four on 1% of the data frame samples then we calculated the silhouette score to pick best k and the algorithm gives better silhouette scores. As show below the algorithm gives better silhouette scores is K-Means and the greatest silhouette score achieved when k = 2 as shown in figure 9 and 10.



```
##### Number OF Clusters 2
Predictions
silhouette score =  0.6350199552565023
##### Number OF Clusters 3
Predictions
silhouette score =  0.5930354554119541
##### Number OF Clusters 4
Predictions
silhouette score =  0.574748929194361
```

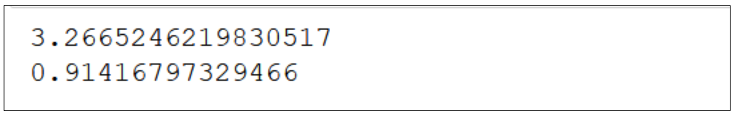
Figure 9: silhouette scores for K-Means for k equal 2, 3 and 4



```
##### Number OF Clusters 2
silhouette score =  0.05141541303165487
##### Number OF Clusters 3
silhouette score =  0.025590718767906348
##### Number OF Clusters 4
silhouette score =  -0.13987418428070314
```

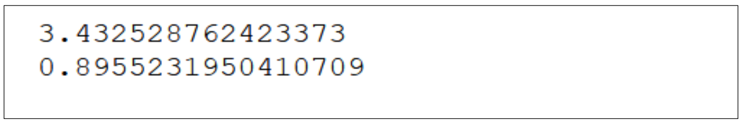
Figure 10: silhouette scores for GMM for k equal 2, 3 and 4

  Then we tried four regression techniques Linear Regression, Polynomial Regression, Lasso and Decision Tree and the most powerful and suitable for the date set was the decision tree. We applied regression on highly contributed features for calculating ride total amount Milage', 'Duration', 'IsWeekday', 'SurchargeAmount', 'OriginLatitude', 'OriginLongitude', 'DestinationLatitude' and 'DestinationLongitude'. We evaluated models using two metrics mean square error and r2 score.



```
3.2665246219830517
0.91416797329466
```

Figure 11: Mean square error and r2 score for cluster 1 regression model



```
3.432528762423373
0.8955231950410709
```

Figure 12: Mean square error and r2 score for cluster 2 regression model

- Regression Model without clustering:

  We applied the decision tree regression technique overall data.

Evaluation

We compared the two approaches we mentioned before and we have chosen the approach give better regression results, which is approach one, as the main purpose for this project is to predict total amount of the ride.

Final Process

The final approach we will follow is, we will provide the clustering model the ride info hour, weekday, isweekday, duration and mileage to cluster the ride to one of two clusters then, we will send the same data to regression model relate to thus cluster to predict total amount of the cluster. We add to the features we use for clustering source and destination latitude and longitude.

Deployment

We used a server instance on Google Cloud platform to deploy our prediction solution.

Our end goal was to have an end-user interface where we can insert the features and we get the prediction results based on the values inserted to predict the fare of the trip. After finalizing our model, we used the pickle operation to serialize our machine-learning model and save the serialized format to a file. We loaded this file on GCP to deserlaize our model and use it to make new predictions. We built a user interface to input the needed features to our model to get the fare predictions.

Deployment Architecture

We used multi-purpose tool App Engine to train our wrapped model, dump its binary representation in Google Cloud Storage, load it from there and deliver predictions using the user interface.

We setup a project on Google cloud platform to accommodate our model deployment.

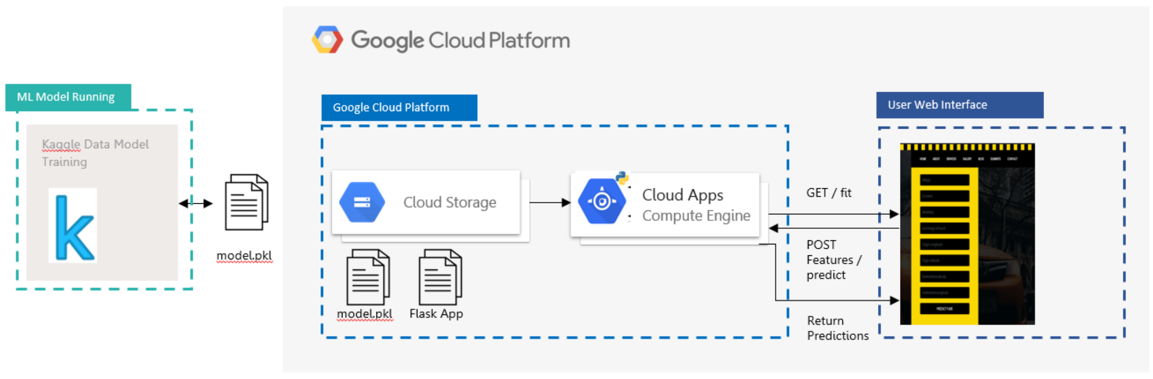
We wrote a flask app (main.py) In particular, it features two handlers /fit and /predict as well as an initialisation handler _load_model.
_load_model checks if a model binary is available at the path specified by environment variables and loads it if available.
fit_model fits a TextClassifier and dumps it to Google Cloud Storage

predict_from_model returns predictions based on text input arguments given that our model is previously loaded in the memory

Desired configuration of the AppEngine service is communicated through the app.yaml file

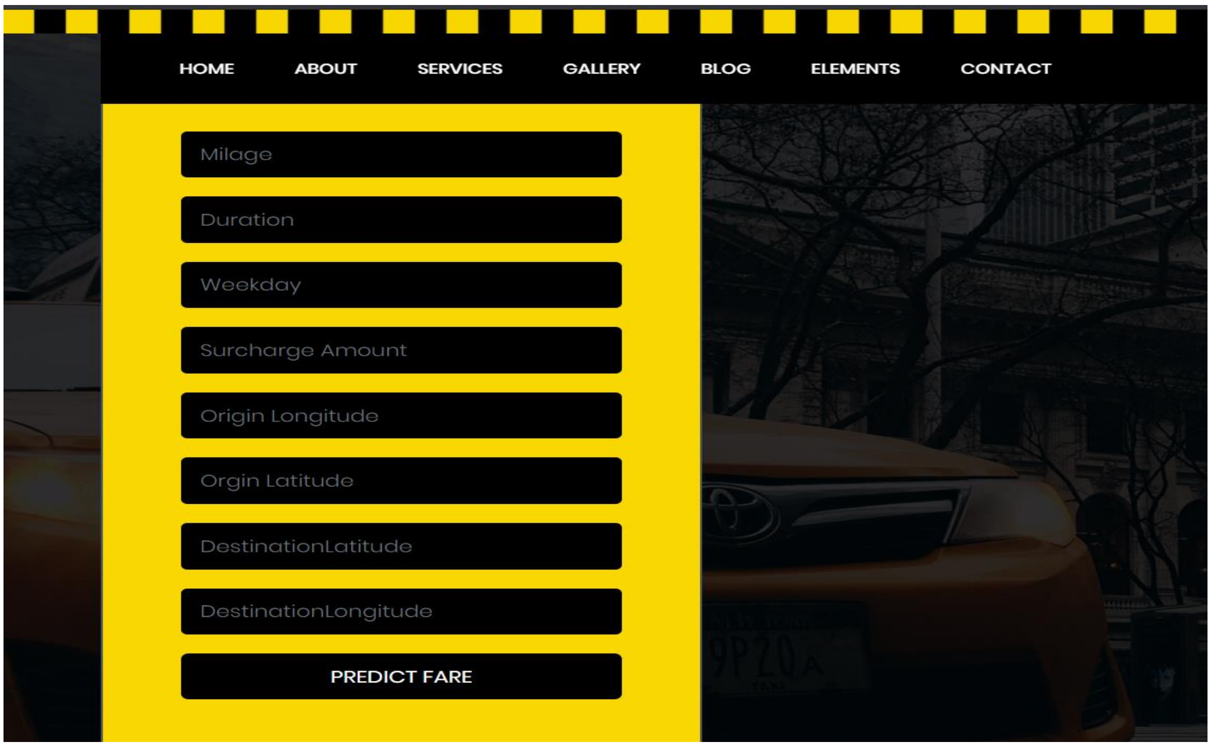
The files needed for deployment:

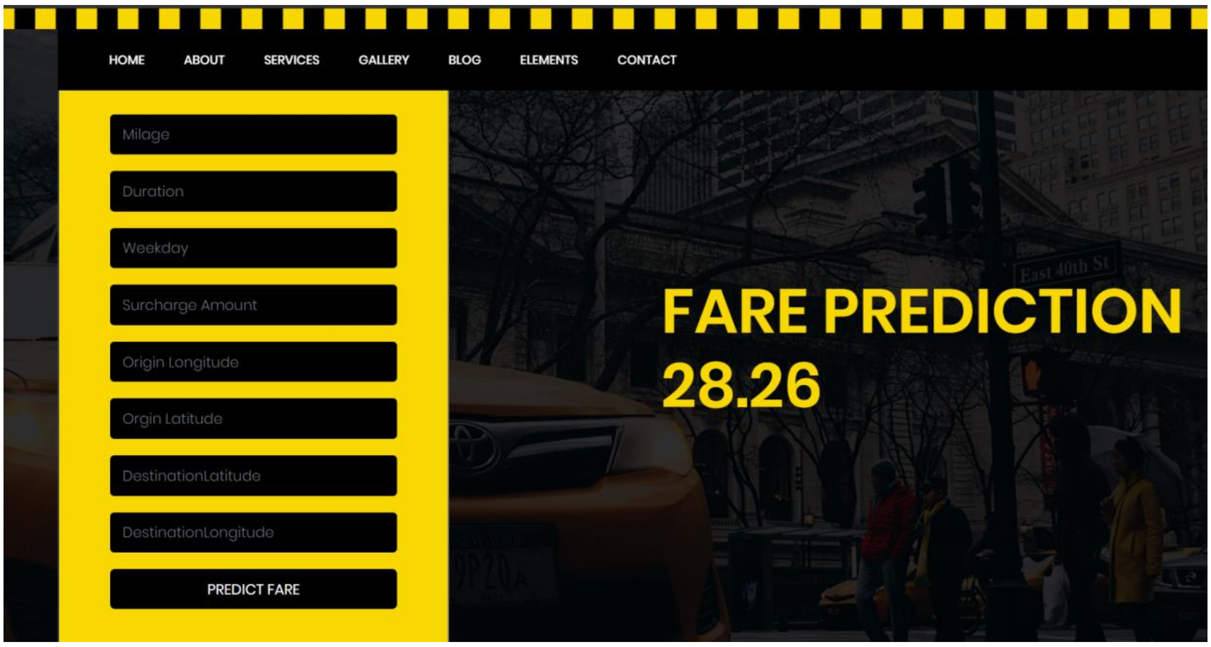
```
|_ main.py # Flask app definition
|_ requirements.txt # Python dependencies
|_ app.yaml # AppEngine config file
```



The user Interface

It's a simple web interface where we can enter the model features and get the fare's prediction using get and post requests to our deployed data model.





FARE PREDICTION
28.26