

Project documentation (Taxi ride fare prediction)

Milestone 1 Report

Initial Intuition

It has been rumored that taxi service providers charge customers more when weather conditions are severe, because customers would be forced to take the cab anyway, our problem is to predict fare prices and state *-based on given data-* whether this rumor is true or not.

Data Cleaning

Null Values

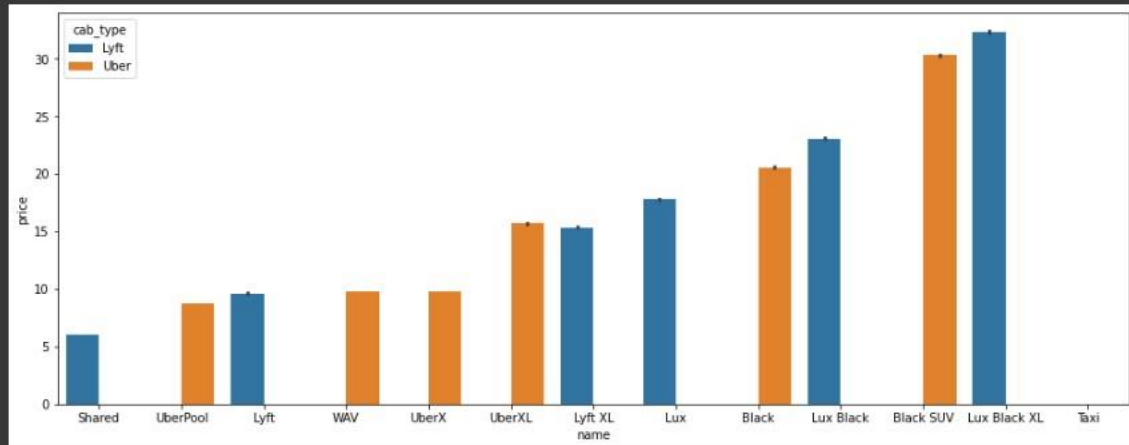
Price

The target feature (“price”) contained null values so the first thought was to drop the rows

```
Null Values in columns
distance           0
cab_type           0
time_stamp         0
destination        0
source             0
surge_multiplier   0
id                 0
product_id         0
name               0
price              44135
```

After further analysis it turned out the all the null values were related to another feature (“name”) which represented the subtype of the ordered cap (eg. UberX, Lyft Lux, etc..).

All the null values were from the “**Taxi**” subtype.



Apparently all *price* values of *Taxi* are missing, could all the missing values from *price* be from the *taxi* cab type? we need to verify this

There are 44135 price null values with Taxi as subtype from a total of 44135 : 100.0%

The final decision was to **drop the rows**, however in [later sections](#) there was an imputing attempt.

Rain

In the “rain” feature the assumption was that the value it contained represented the amount of rain in inches and since rain was null in almost all the weather rows (**85%**) so there was doubt that maybe the null values represented no rain since there were no values of 0 in the rain feature.

Looking further into the subset of rows with null and non-null values

Rows with null rain value statistics						
	temp	clouds	pressure	rain	humidity	wind
count	5382.000000	5382.000000	5382.000000	0.0	5382.000000	5382.000000
mean	38.461557	0.633618	1008.92490	NaN	0.747293	6.375199
std	6.169892	0.312919	13.46695	NaN	0.124451	3.542713
min	19.620000	0.000000	988.25000	NaN	0.450000	0.290000
25%	35.022500	0.400000	997.39000	NaN	0.660000	3.190000
50%	39.360000	0.680000	1009.56000	NaN	0.730000	6.210000
75%	41.470000	0.930000	1020.95000	NaN	0.850000	9.627500
max	55.410000	1.000000	1035.12000	NaN	0.990000	18.180000

Rows with non-null rain value statistics						
	temp	clouds	pressure	rain	humidity	wind
count	894.000000	894.000000	894.000000	894.000000	894.000000	894.000000
mean	42.876644	0.943624	1005.557405	0.057652	0.864474	9.377103
std	2.951239	0.146752	7.841618	0.100758	0.093957	3.066719
min	34.100000	0.100000	988.910000	0.000200	0.520000	1.740000
25%	41.482500	0.970000	999.750000	0.004900	0.860000	7.660000
50%	43.820000	1.000000	1005.945000	0.014850	0.900000	9.890000
75%	44.510000	1.000000	1011.237500	0.060925	0.920000	11.187500
max	51.240000	1.000000	1021.790000	0.780700	0.980000	17.160000

It can be noticed that

- Rows with null rain has **lower** average of clouds and winds
- The distribution of clouds compared with the distribution of the non-null values has lower values in the IQR

Thus it can be inferred that the “rain” rows with null values can be imputed with 0 since they represent **no rain**.

Redundant Features

“product_id” and “name”

At first glance the “product_id” and “name” features seemed to represent the same thing, if that was the case one of them had to be dropped since they presented no new information.

Value counts of 'product_id' feature

55c66225-fbe7-4fd5-9072-eab1ece5e23e	44204
8cf7e821-f0d3-49c6-8eba-e679c0ebcf6a	44135
997acbb5-e102-41e1-b155-9df7de0a73f2	44108
6d318bcc-22a3-4af6-bddd-b409bfce1546	44038
9a0e7b09-b92b-4c41-9779-2ad22b4d779d	44024
6f72dfc5-27f1-42e8-84db-ccc7a75f6969	43983
6c84fd89-3f11-4782-9b50-97c468b19529	43977
lyft_plus	41142
lyft_lux	41041
lyft	41015
lyft_luxsuv	40979
lyft_premier	40969
lyft_line	40841

Value counts of 'name' feature

UberX	44204
Taxi	44135
UberPool	44108
Black SUV	44038
WAV	44024
UberXL	43983
Black	43977
Lyft XL	41142
Lux Black	41041
Lyft	41015
Lux Black XL	40979
Lux	40969
Shared	40841

By looking into the counts of unique values in each feature it became clear that they did represent the same thing, it seems that product_id was meant to be used in the system internally since some of the names have hash-like values.

So the **“product_id”** feature was **dropped** since the **“name”** feature was more interpretable to the development team.

Other Features

Features like **“id”** were also dropped since they didn't present any information on the problem

Also **“date”**, **“location”** and **“timestamp”** were dropped after proper preprocessing in the [later sections](#).

Data Preprocessing

Encoding Timestamp to date

Encoding the timestamp up to the hour produced null values when joining the datasets which is probably due to the cab orders not having a weather reading at the time.

So the hour value was discarded and the timestamp was only taken up to the day.

Timestamp feature was dropped since it was replaced with **“date”** feature

Joining Datasets

After encoding the timestamps of both the **“taxi_rides”** dataset and **“weather”** dataset a merge was done on them so that all data can be present in one dataframe.

Merging was done based on the **“date”**, **“source”** from the **“taxi_rides”** and **“date”**, **“location”** from the **“weather”**

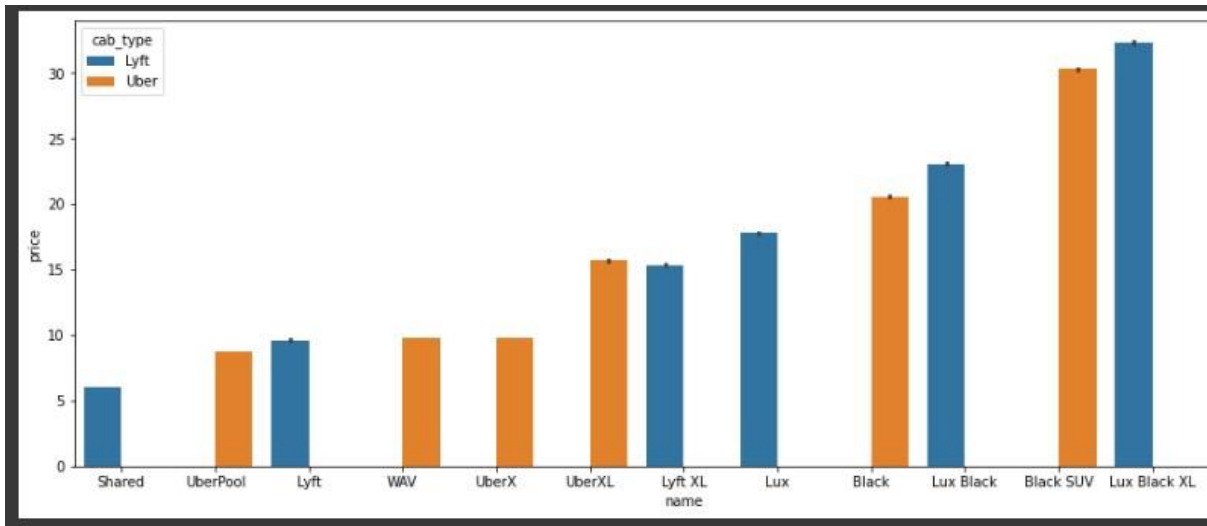
Use of **“source”** only when joining was based on reasoning that if the weather did affect the price, the weather of the pick-up location would be the deciding factor, not the arrival. location.

Encoding

“Cab_class”

This feature was the jackpot, the sk-learn LabelEncoder encoded the **“name”** (which to remind contained the subtype of the cab) without consideration that different

subtypes have different price ranges, keeping this in mind, the name feature was encoded such that subtypes of the same price range have the same label. We tried to impute the nulls in our target (“price”) to keep taxi in name cab-class by set taxi class to category 1 and set value of nulls by class 1 average price, but it didn’t affect the model behavior, so we dropped the nulls.



Using the above graph the subtypes were grouped and encoded into 5 groups ranging from economy to premium class.

This decreased the model’s MSE from **42.05** to **8.27**!

Other Non-Integer Features

Features such as “cab_type”, “destination”, “source” were label encoded using the sk-learn default LabelEncoder.

There were trials to one-hot encode the destination and source however there wasn’t a noticeable change in the models performance so they were discarded to avoid the exploding dimensionality produced by the one-hot encoding.

Feature Engineering

Rain Feature

Referring to google:

Light rainfall is considered **less than 0.10 inches** of rain per hour. Moderate rainfall measures **0.10 to 0.30 inches** of rain per hour. Heavy rainfall is more than **0.30 inches** of rain per hour.

0 : no rain

1 : light rain

2 : mid rain

3 : heavy rain (doesn't exist in the data)

According to the above information, rain feature was encoded to one of 3 categories No rain, light rain, mid rain.

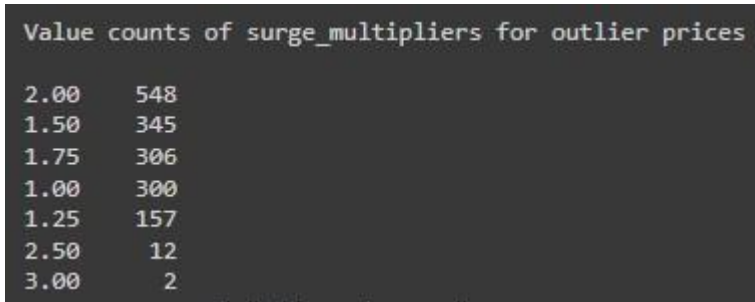
Clouds Feature

Making the assumption that clouds are on normalized [Okta Scale](#) that means values *less than 0.1 are sunny days*, and if weather did affect the price then sunny days would see a decrease in price since people might prefer to walk or bike shorter distances than take a cab.

So values of clouds less than 0.1 were encoded to 1 which indicated a sunny day while others were encoded to 0 which meant a cloudy day

Outliers

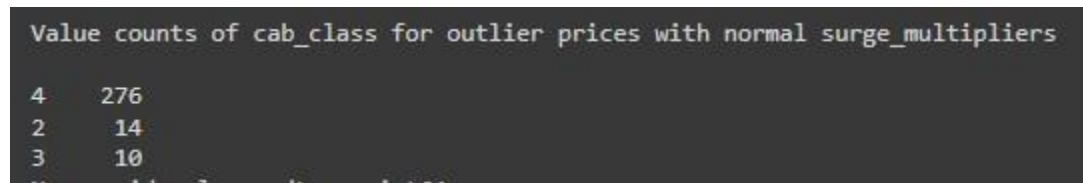
Outliers were checked for in the “price” column using z-score and 1670 prices were found to be outliers, however after looking further it was found that those outliers were correlated to a **high “surge_multipliers”** which would explain their difference from the rest of the prices.



```
Value counts of surge_multipliers for outlier prices
```

2.00	548
1.50	345
1.75	306
1.00	300
1.25	157
2.50	12
3.00	2

There's still 300 rows with normal surge_multipliers



```
Value counts of cab_class for outlier prices with normal surge_multipliers
```

4	276
2	14
3	10

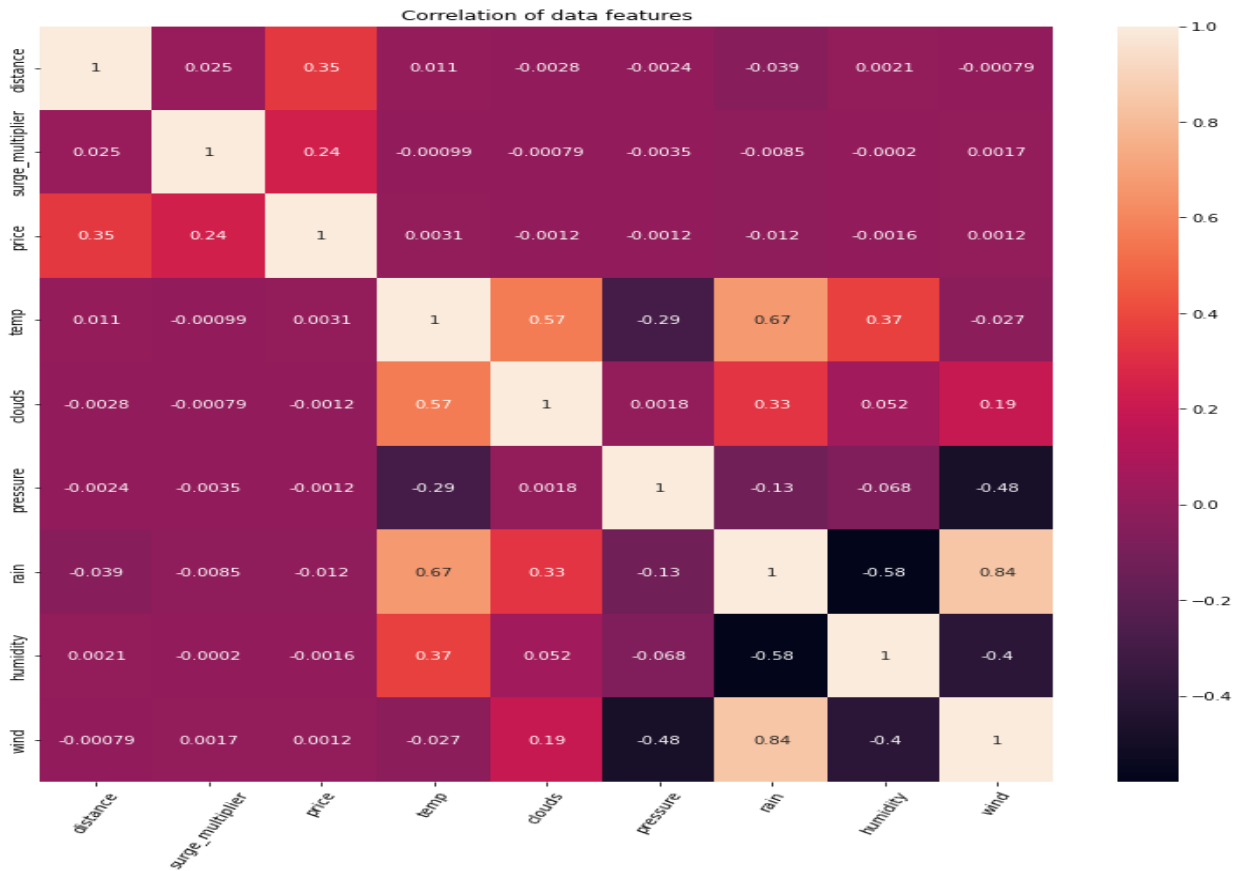
This would explain the high prices, since they are mostly premium types.

Normalization

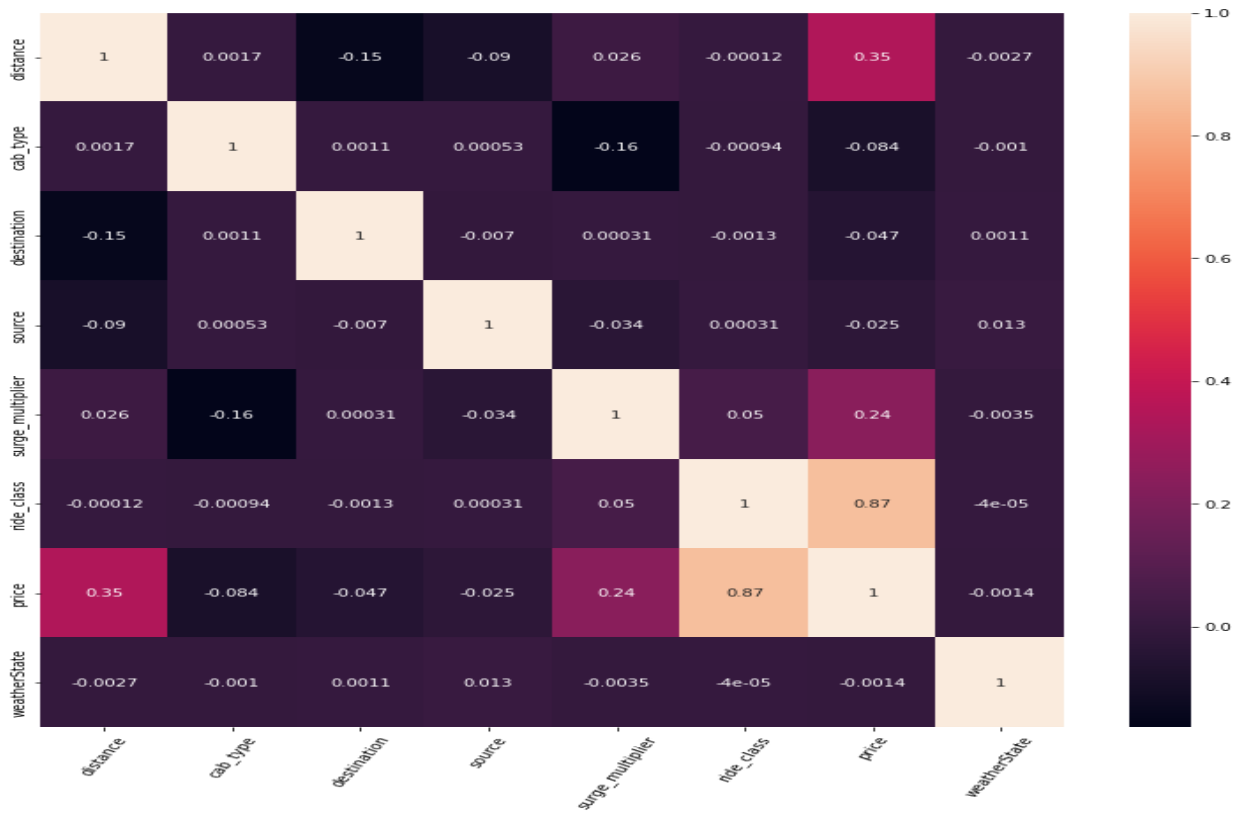
We tried feature scaling using min-max scaler and the model acts as same as without feature scaling, so we didn't apply it to the final model.

Dimensionality Reduction using PCA

We recognized that the correlation between weather features and our target (“price”) is so weak, as shown in the following figure:



So we decided to use dimensionality reduction algorithm called PCA to reduce weather features to one feature called “weatherState”, the following figure represents the correlation after applying the algorithm:



The model gave us the same accuracy.

Applied Models

Linear Model

We used a multivariable linear regression model using Sklearn linearModel.

Train / Validation

It gave us approximately the same accuracy in both train and validation which equals to 90%, the following table explain in detail the model accuracy and mean squared error:

Dataset	Mean square error	R2_Score
train	8.13502792809099	0.9065721071325505
validation	7.948716441361453	0.907798866242228

That's the best model we achieved after feature selection, we tried different models on subsets of features, and the model takes 1 seconds to train.

Cross Validation

We tried k-fold cross validation on our data when k equals 5 using mean squared error as our metric, it gave us average score: 17.613210853408418, and it takes 1 second to train.

Polynomial Model

We used polynomial regression with polynomial degree 3, using Sklearn.

Train / Validation

The model gave us approximately the same accuracy in both training and validation set, it gave us 95%.

Dataset	Mean square error	R2_Score
train	4.067000317682811	0.9532919526114622
validation	3.9766996594231956	0.9538722736031894

That's the best model we achieved after applying feature selection and different polynomial degree, and the model takes 6 seconds to train.

Cross Validation

We tried k-fold cross validation on our data using polynomial degree equal 3 when k equals 5 using mean squared error as our metric, it gave us average score: 6.13082404437927, and it takes 33 seconds to train.

Milestone 2 Report

Data Preprocessing

For milestone 2 data preprocessing was pretty much the same as milestone 1.

[Found above.](#)

Applied Models

Linear Logistic

We used multivariate logistic regression model using Sklearn.

Train / Validation

It gave us approximately the same accuracy in both train and validation which equals to 83%, the following table explain in detail the model accuracy and F1_Score:

Dataset	Accuracy	F1_Score
train	0.8351421666320982	0.8351421666320982
validation	0.8340006492803809	0.8340006492803809

Polynomial Logistic

We used polynomial logistic regression with polynomial degree 3, using Sklearn.

Train / Validation

The model gave us approximately the same accuracy in both training and validation set, it gave us 85%.

Dataset	Accuracy	F1_Score
train	0.8508693221271338	0.8508693221271338
validation	0.8504851567290698	0.8504851567290698

Decision Tree

We used decision tree classifier, using Sklearn.

Train / Validation

The model gave us approximately the same accuracy in both training and validation set, it gave us 86%.

Dataset	Accuracy	F1_Score
train	0.8682151842800588	0.8682151842800588
validation	0.864417631569455	0.864417631569455

Random Forest (ensemble learning)

We used random forest classifier, using Sklearn.

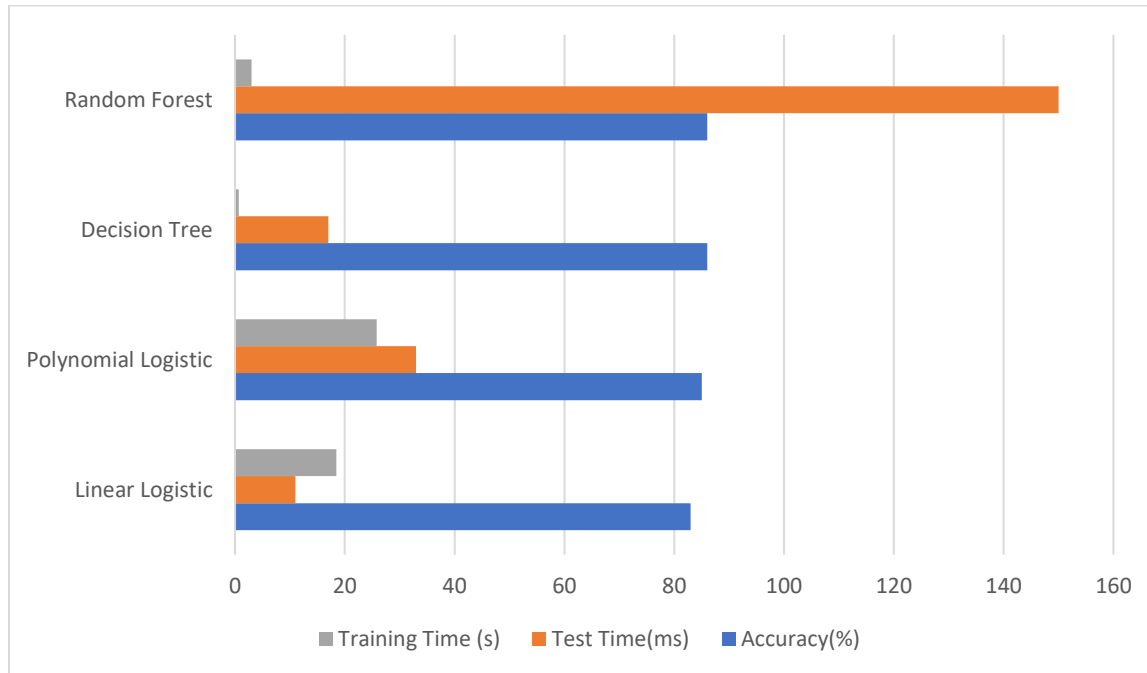
Train / Validation

The model gave us approximately the same accuracy in both training and validation set, it gave us 86%.

Dataset	Accuracy	F1_Score
train	0.8676538222218214	0.8676538222218214
validation	0.8642102225588861	0.8642102225588861

Summary

Below we find a three bar graph that summarize classification accuracy, training time and test time for all the models we made



Hyperparameter Tuning

Randomized Search using Sklearn's *RandomizedSearchCV* was attempted to tune the hyperparameters

Linear Logistic & Polynomial Logistic

Hyperparameter tuning didn't affect the logistic models that much as the accuracy was almost the same using the default parameters

The final parameters used were

```
{'C': 0.47469656914931324, 'penalty': 'none', 'solver': 'lbfgs'}
```

Decision Tree

Using manual parameters the model was overfitting, getting 89% accuracy on the training data but 84% on the validation set

However after tuning the parameters the model was able to generalize better getting **86%** on both the training and validation

The best parameters found were

```
{'criterion': 'entropy', 'max_depth': 30, 'min_samples_split': 501}
```

Random Forest

Similar to the Decision tree the model was overfitting the training set with similar scores

After tuning the hyperparameters the model was able to achieve **86.5%** validation accuracy

The best parameters found were

```
{'criterion': 'entropy',  
 'max_depth': 30,  
 'min_samples_split': 201,  
 'n_estimators': 50}
```

Project Conclusion

According to our analysis of the given data, there is little to no correlation between weather severity and taxi fare prices.