2017

Capstone Project Report- Machine Learning Engineer Nanodegree v0.3

5th August 2017

Hrishekesh Shinde (hrishekesh@gmail.com)

Contents

1. Context

This document provides a report for the capstone project of machine learning engineer nanodegree offered by Udacity. The problem selected for capstone project is a live competition to predict travel time in New York using taxis. The link for this project on Kaggle is as follows: https://www.kaggle.com/c/nyc-taxi-trip-duration this report explains the details of solution approach used for this project along with the conclusions and possible improvements.

2. Definition

a. Project Overview

This project attempts to predict the time required for a New York City taxi to travel from one location to another based on data available from previous trips. This project utilizes a large data set which is available on Kaggle as this is an active competition. In the data exploration section in this report I have provided more insight in the available dataset.

b. Problem Statement

Given the available dataset for large amount of cab travel details in the city of New York, in this project I will attempt to

- Predict the time duration required for a taxi based on the pick-up location, drop-off location and the date and time of pick-up.
- Model the dataset to derive new features from the given features in the dataset

c. Metrics

Two metrics which are used to evaluate this project are

- R2 evaluation metric which is available out of the box from sklearn library
- RMSLE – Root mean square logarithmic error. The details for this metric are available at https://www.kaggle.com/c/wikichallenge/details/Evaluation

The details of the chosen metrics is as follows.

Metric	Justification for choosing this metric
RMSLE	This metric provides the root mean square log error for the model – this ensures that negative and positive errors do not cancel each other and provides a good idea of how accurate the predictions can be.
R2	A short coming of RMSLE metric is we cannot evaluate in case the model does not fit correctly on the training data. The score of R2 metric suggests whether the model generated fits appropriately – Negative values indicate that the fit may not be correct

3. Analysis

The dataset has 1458644 rows will be used to model and predict the results. The data set provides the following 11 features:

- Id
    - This column represents a unique row id in the dataset.
    - This column not be used as a feature to model the data

- vendor_id
  - This data set is distributed with two different vendors. This feature helps us distinguish between the two vendors
- pickup_datetime
  - This feature will provide the time and date of pick up
  - The format in which this is available is YYYY-mm-dd HH:MM:SS
  - For modelling purposes, this date will be converted to float value
- dropoff_datetime
  - This feature represents the drop off date and time and is similar to pick up date
  - I think this feature is redundant because the difference between drop-off date time and pick-up date time will provide the information on trip duration. Hence this feature will be discarded.
- passenger_count
  - Represents the number of passengers for the trip
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude
  - As the names suggest the above 4 features represent the latitude and longitude information of pick up and drop off locations.
- store_and_fwd_flag
  - Indicates whether the trip record was held in vehicle memory before sending to the vendor.
- trip_duration
  - Travel time in seconds.

a. Data Exploration - Derived Features

Based on the features available, I can derive more features which will provide more precise information. The list of all features derived and how they can be potentially used to generate a model is provided below:

Derived feature	Existing feature used	Justification to separate this feature	How is this feature used
Pickup_weekday	Pickup_datetime	The trip duration can vary based on the day of the week. For example, on a particular day the traffic can be more based on whether it is a holiday or a working day	This feature is used to create model
Pickup_hour	Pickup_datetime	The trip can vary if we travel during the peak hours. For example travelling between 12 AM to 6 AM may be faster than travelling at 8 AM	This feature is used to create model

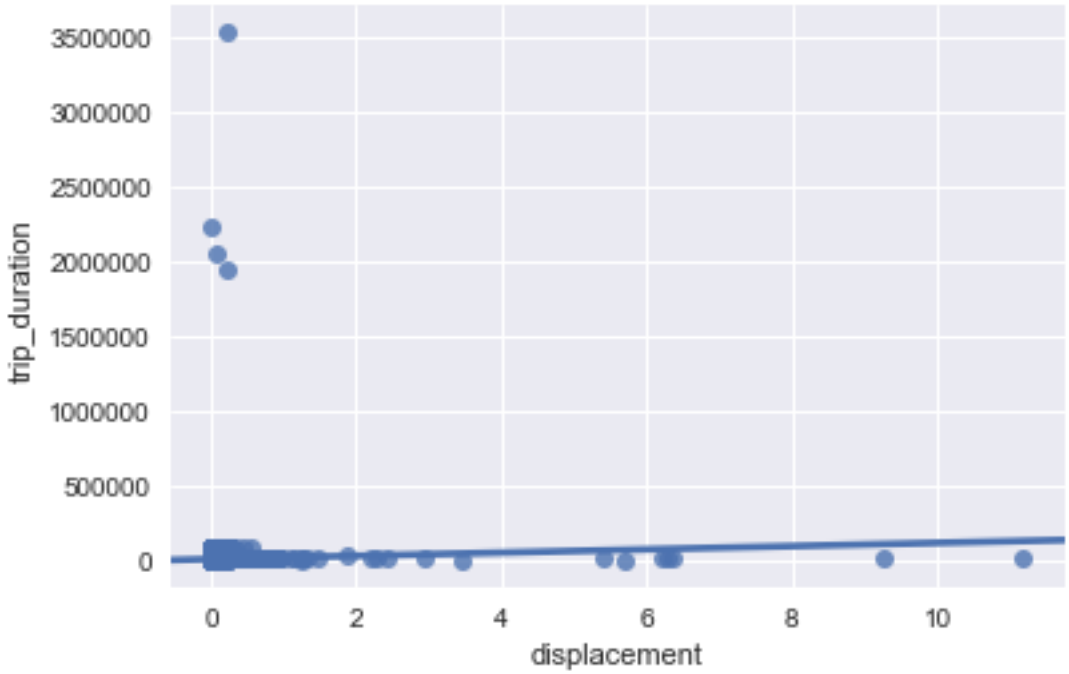
Pickup_datetime_float	Pickup_datetime	Pick up date time cannot be directly used to train a model as it is a string object. Hence I have converted it to float using the time library followed by taking the natural log using the math library	This feature is used to create model
Store_fwd_flag_encode	Store_and_forward_flag	This flag is encoded to use it to train the model	This feature is used to create model
Trip_duration_hrs	Trip_duration	Converted trip duration in seconds to hours to round off precision errors	This feature is used to create model
Latitude_difference	Pickup_latitude, dropoff_latitude	This is to calculate the approximate distance traversed and calculated as the difference between pickup and drop-off latitudes	This feature is used to create model
Longitude_difference	Pickup_longitude, dropoff_longitude	This is to calculate the approximate distance traversed and calculated as the difference between pickup and drop-off longitudes	This feature is used to create model
Displacement	Latitude_difference, longitude_difference	This is the approximate distance travelled which is calculated using square root of sum of squares of latitude and longitude differences. Following function is used for the same: math.sqrt(math.pow(lat_diff,2) + math.pow(long_diff,2))	This feature is used to create model
Speed	Displacement, time_duration_hrs	This feature was considered for data analysis and outlier detection.	This feature is used for data analysis only as we will not be able to derive it unless we know the travel duration

After addition of new features, only the following 13 features will be retained to build the model:

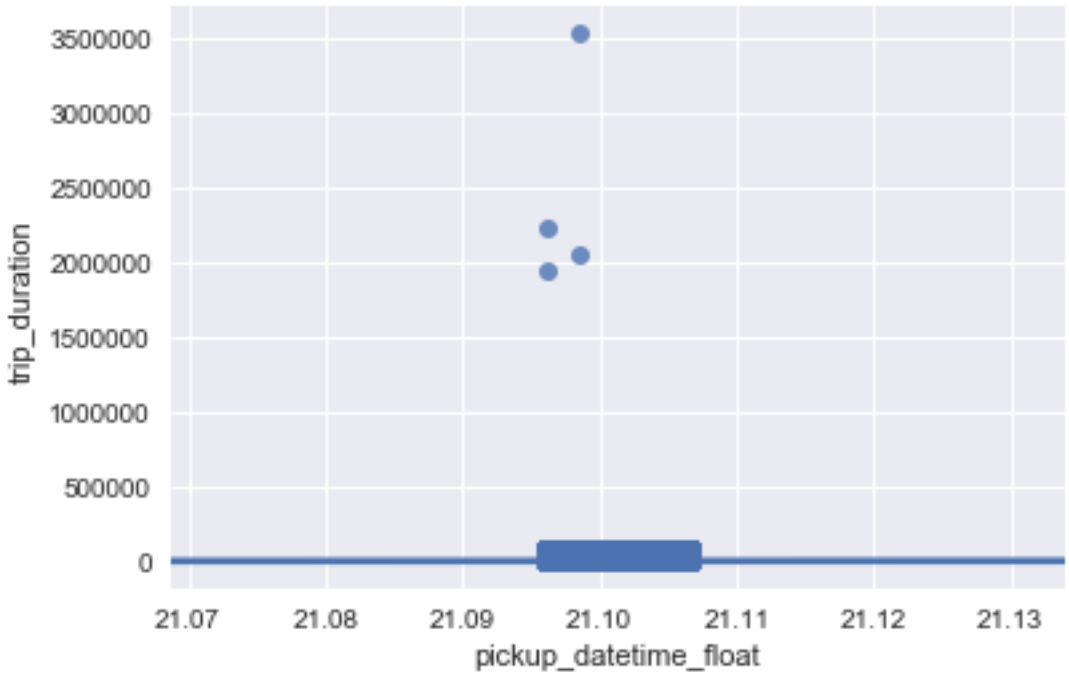
vendor_id, passenger_count, pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude, pickup_weekday, pickup_hour, pickup_datetime_float, store_and_fwd_flag_encode, latitude_difference, longitude_difference, displacement

b. Exploratory Visualization

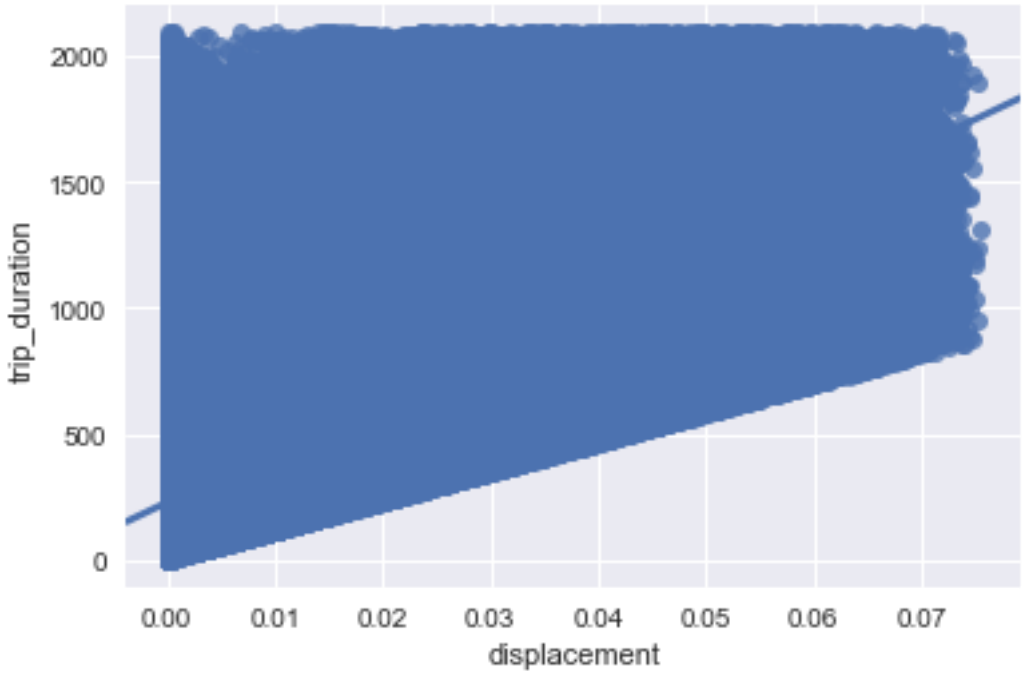
Based on the derived features, I have explored the following. Please note that all the graphs are not included in the report. Following graph shows the relationship between the displacement and trip duration.

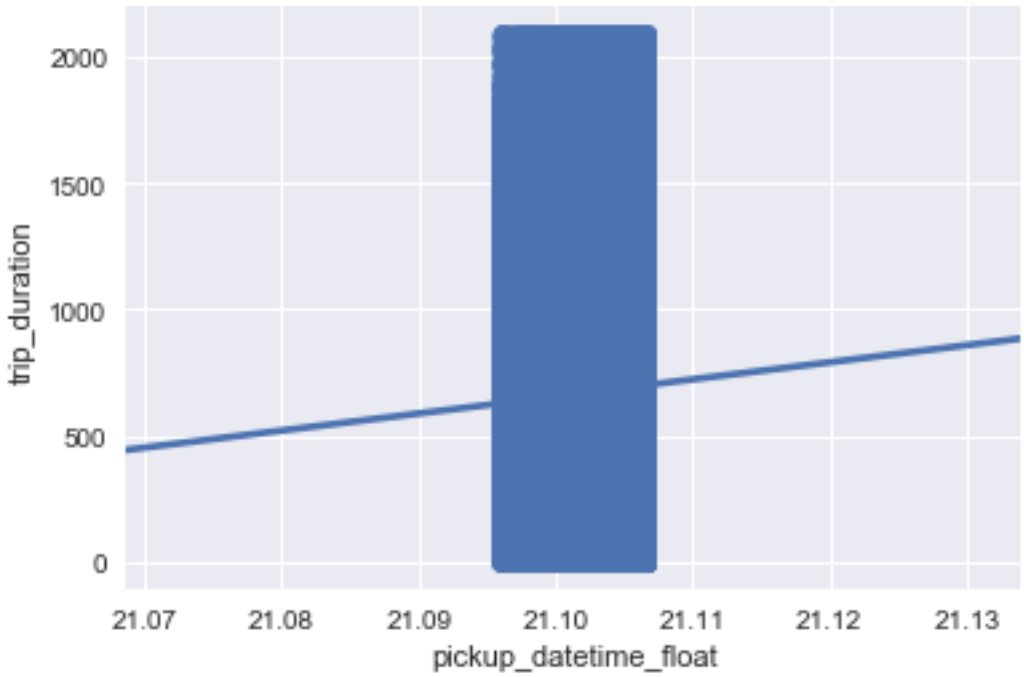


Similarly the time duration based on pick up date time on the available data is as below:

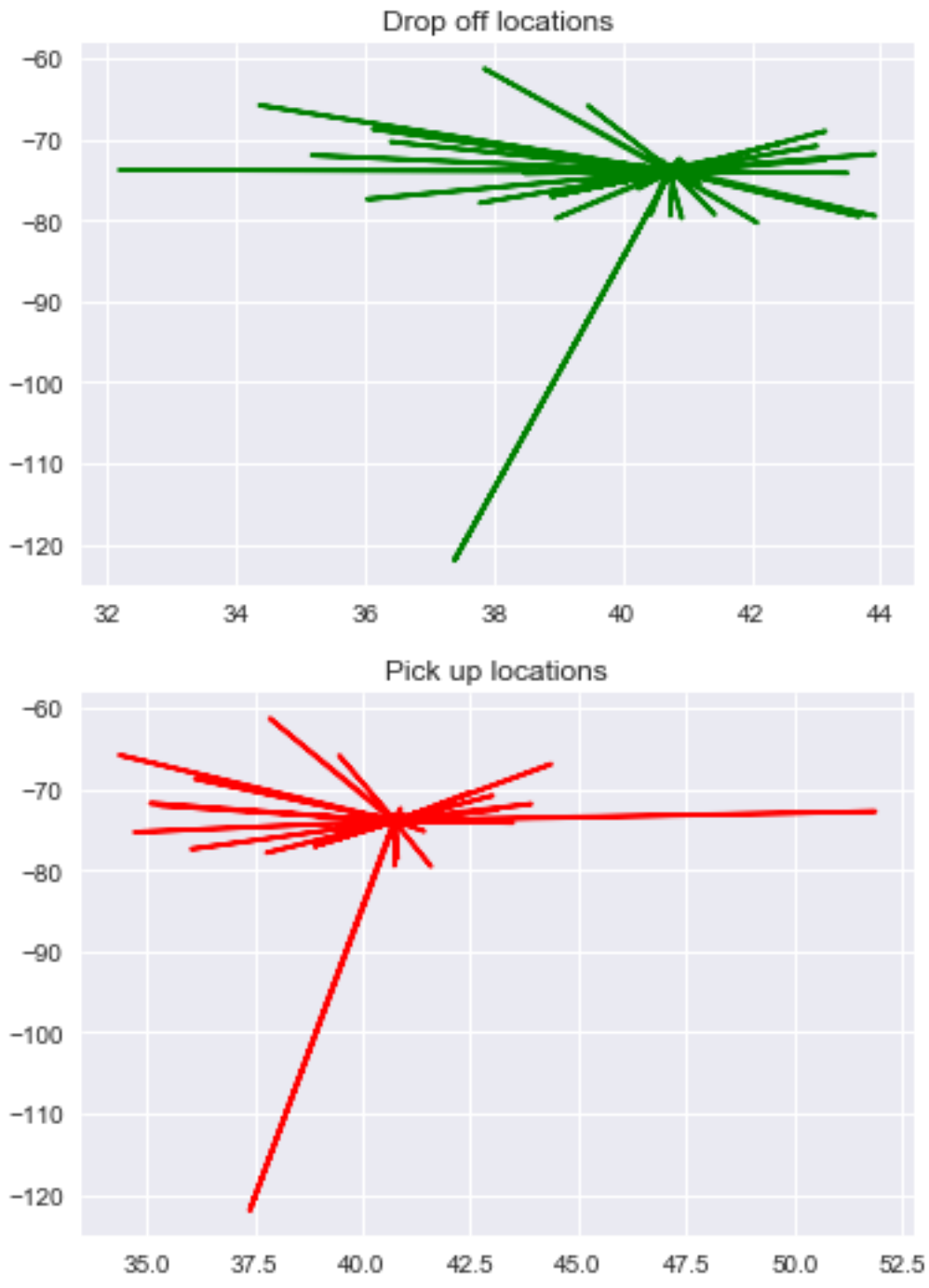


Both these graphs clearly indicates outliers present in the data. Hence I have performed the outlier analysis and eliminated the outliers from the available dataset. After eliminating the outliers, the dataset was reduced to 1192409 rows which means that I have eliminated about 19.3% of available data. The relationship between various features and trip duration is as shown below for the good data:





Following graphs represent the pick-up and drop-off locations based on latitudes and longitudes.



c. Algorithms and techniques

Following techniques and libraries were used for data exploration;

Data exploration step	Library used	Technique used	Justification to carry out this step
Import data	Pandas	Out of the box library feature	-
Derive new features	-	Simplify complex relationship between data features and labels by logically splitting complex data	To get more clarity and explore the relationship between the features and labels

Outlier Analysis	Numpy	Calculate inter quartile range on the data and retain data which falls within 1.5 times interquartile range	Eliminate outliers to prevent the model getting skewed.

Following Regressors were used to model the test data:

Model Used	Characteristics why this model was chosen
Decision Tree Regressor	• Decision trees can predict the value of target variable by learning simple decision rules from the target data • For this project I think decision trees will be a good fit because the time duration for taxi trip will be primarily a function of distance travelled and speed at which this distance is travelled. • I chose this regressor as the training time for this model is substantially low as compared to other models. This model tends to over-fit however with adequate performance tuning, this model produced the best results
MLP Regressor	• MLP uses a multilayer perceptron that utilizes backpropagation without activation in the output layer. I expected this model to produce better results for this project as this model will not over-fit like the DecisionTreeRegressor. • MLP internally uses SGD to reduce the root mean square error hence this may result in better RMSLE scores • I chose this model because I wanted to use a neural network based model in regression. This model initially produced the best results however because it took too long to run, I could not optimise it further • This model yielded negative time predictions for some data in test results
SGDRegressor	• This model is chosen because this model tends to reduce the Root mean square errors • Did not explore this model in detail as it produced substantially high errors and a large negative R2 score

The techniques which I tried to use for optimization in this project are as below:

- ✓ Simplify complex features into simple ones – like pick up time was split into pickup hour and week day
- ✓ GridSearchCV – This algorithm took too long to run hence was abandoned
- ✓ Divide and conquer algorithms to tune the model parameters as GridSearchCV and RandomizedSearchCV could not be used. A more detailed discussion on this is provided in section Implementation and Refinement under model tuning
- ✓ Bagging and boosting to tune the models

d. Benchmark

Ridge regressor model was chosen as the benchmark model of this project. The reason for choosing this models is that the model is fast and will provide a good benchmark to start with Methodology

4. Methodology

The methodology used for this project can be summarized as below:

- Derive new simple features from existing complex features
- Eliminate outlier data
- Create a benchmark model
- Model data for set of relevant features based on principal component analysis
- Split the data using either train test split or shuffle split
- Identify regression models – Decision tree regressor and MLP regressor were be used for this project
- Optimise the parameters of the model
- Apply bagging regressor over the chosen regressor
- Apply boosting regressor using Adaboost regressor

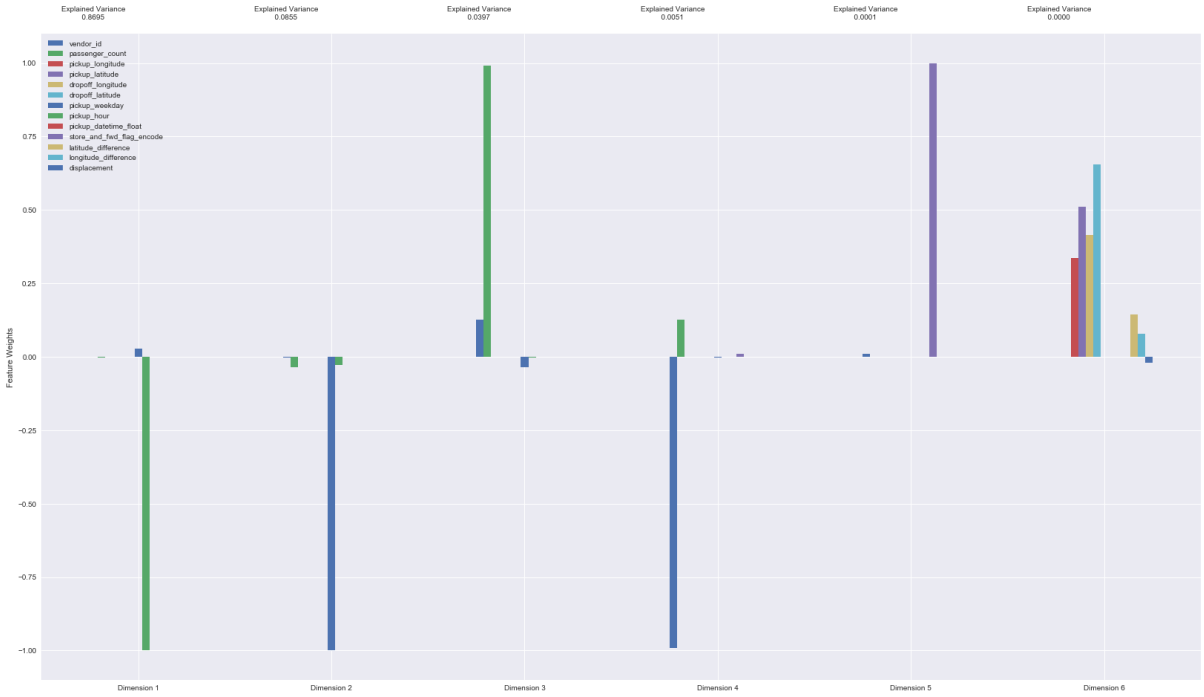
The list of libraries used along with the versions are provided in section 8 of this document

a. Data Pre-processing

Pre-processing done on the data is explained in detail in Data exploration – derived data section. The pre-processing done using the following techniques:

Feature	Technique used for pre-processing
Pickup_datetime	Split the data into features of weekday and pickup hour
Store_and_fwd_flag	Encode the flag to numerical value
Derived features	Add more features to build a robust model
Drop features which are not required	Eliminate duplicate or redundant features based on PCA – As per PCA, I have dropped the features for pick up and drop off latitudes and longitudes. The derived features were identified as the principal components. The graph below represents the PCA analysis done.
Split data into training and testing data	In this project, I have used both techniques – test train split and shuffle split.

PCA results considering 6 components are as shown below



b. Implementation and Refinement

I have used multiple algorithms and techniques to train the models and evaluate the models on training and testing data. The detailed steps are as provided below:

- The data was split into training and testing labels using train_test_split module
- The benchmark model was selected as Ridge with random_state=None
- The regression models which were chosen are as below. The reasoning why these models were chosen is provided in section 3c:
    - Decision tree regressor
    - MLP regressor
    - SGD regressor
    - XGBoost regressor – This is a popular model on Kaggle community. However like MLP regressor, XGBoost regressor took too long to tune.
- Of the above models selected, SGDRegressor had a negative R2 score and a very large RMSLE error hence this model will not be further investigated
- Model selection
    - DecisionTreeRegressor model had a score lower than benchmark score however the RMSLE score was close to the benchmark score
    - MLPRegressor model had a better R2 score than benchmark score however slightly higher RMSLE score
    - Both models were further investigated and tuning will be attempted
- Model Tuning
    - Techniques used to optimise the model are as described below:

Model Optimization technique	Justification	Challenges faced	Alternative Solution used
Grid Search CV	This technique was chosen to find the best parameters for	Could not use this technique as it took too long to run based on	Randomized Search CV technique was considered however I decided to go ahead with manually

	DecisionTreeRegressor and MLPRegressor.	the machine configurations	tuning the parameters over loops for a selected set of parameters. For DecisionTreeRegressor following parameters were tuned using the divide and conquer technique Max_depth, Max_leaf_nodes, Min_samples_leaf For MLP Regressor following parameters were tuned again using divide and conquer technique Hidden_layer_sizes, alpha, max_iter
Bagging and Boosting	This technique was chosen to optimize the results obtained from the chosen model.	Because of the machine configurations, these techniques took too long to run.	I have applied Bagging and Boosting separately over DecisionTreeRegressor however I could not apply boosting over bagging. These techniques could not be used for MLPRegressor as it took too long to run.

  - Grid search CV was attempted using ShuffleSplit to find the best model on DecisionTreeRegressor to find the best estimator parameters. However with the machine configurations available this took too long to run and hence was aborted.
  - DecisionTreeRegressor was manually tuned with the following parameters. This tuning increased the R2 score from 0.25 to 0.61
    - Max_depth =29
    - Max_leaf_nodes = 10000
    - Min_samples_leaf = 150
  - BaggingRegressor was utilized to optimize the DecisionTreeRegressor further. This increased the R2 score to 0.629
  - Boosting was attempted over BaggingRegressor using AdaBoostRegressor. However this took a long time to run hence was aborted.
  - MLPRegressor was also used however this did not have a better R2 or RMSLE scores as compared to BaggingRegressor
- I have also attempted XGBoost regressor however the best results were produced by bagging over decision tree regressor. XGBoost also had negative numbers in the predictions like MLP Regressor

c. Complications and challenges faced during implementation

Following are the challenges faced during implementation and the details about the alternatives used

Implementation phase	Challenge	Technique used
Benchmarking	When I ran the regressor models on the raw data, I got R2 scores within the range 0.001 to 0.005	Split complex features into simple ones to derive new features. This increases the R2 score substantially to about 0.48
Data Analysis	During data analysis, the outliers were predominant. The visualizations were also skewed because of the outlier data	Used standard deviation technique to eliminate all data which lies outside 1.5 times the interquartile range
Model tuning	Grid search CV took too long to run	Applied Divide and conquer technique to tune the model parameters
Model tuning	Ensemble methods of bagging and boosting could not be applied because of low machine configurations	Could not improve on this as it requires a machine with better configurations.
Model tuning	The models seemed to perform better on validation set and performed badly on testing set.	This is a typical challenge because of overfitting. I used the principal component analysis technique to recognise only the required features and trained the model only on these set of features. This technique reduced overfitting in the models

5. Results

a. Model evaluation and validation

The results for R2 score and RMSLE score are provided below

Model	Split Type	Tuning parameter	R2 Score	RMSLE score
Ridge (benchmark)	Train_test_split	None	0.4833	0.0627
DecisionTreeRegressor	Train_test_split	None	0.2538	0.0751
SGDRegressor	Train_test_split	None	-1.372	14.011
MLPRegressor	Train_test_split	None	0.5266	0.1126
DecisionTreeRegressor	Shuffle_split	Grid Search CV - Parameter tuning abandoned because it too long to run	NA	NA
DecisionTreeRegressor	Train_test_split	max_depth=29, min_samples_leaf=150, max_leaf_nodes=10000	0.6183	0.0538
AdaBoostRegressor	Train_test_split	n_estimators=100	0.5171	0.0621
BaggingRegressor	Train_test_split	n_estimators=100	0.6293	0.0530

MLPRegressor	Train_test_split	hidden_layer_sizes=(100,), alpha=0.0001, max_iter=200	0.5621	0.1110
XGBoostRegressor	Train_test_split	None	0.6042	0.0548
XGBoostRegressor	Train_test_split	max_depth=10	0.6450	0.1136

MLP Regressor has produced good R2 score and RMSLE score however has produced negative values in prediction data. Hence these results could not be used.

Bagging and boosting could not be applied together because of limitations on the machine configurations

I have tried also tried to train model with above regressors without adding the derived features. The R2 score was approximately 8 to 10% lower with DecisionTreeRegressor and MLPRegressor for data without derived features.

The benchmark R2 score of 0.48 was improved up to 0.63 using BaggingRegressor over DecisionTreeRegressor however this improvement is limited by the machine configurations. I believe this can be further improved if we apply bagging and boosting over MLPRegressor.

6. Conclusion

In this project I have explored the impacts of the following:

- Data pre-processing
  - When raw data was used without any pre-processing, the models gave a very large error on the validation data set
  - The errors drastically reduced as I started to split some of the complex features into simple features. This was particularly interesting as the models were able to derive better relationships on simple data
  - The models were further improved when outliers were eliminated.
  - Feature reduction did not have an impact on the model however had slight improvement on the training time
- Define performance metrics
  - I have used the R2 score metric and RMSLE metric to evaluate the models. The justification why these models were chosen are provided in section 2c
- Model selection
  - Decision tree regressor – I chose this regressor as the training time for this model is substantially low as compared to other models. This model tends to over-fit however with adequate performance tuning, this model produced the best results
  - MLP regressor – I chose this model because I wanted to use a neural network based model in regression. This model initially produced the best results however because it took too long to run, I could not optimise it further
  - XGBoost regressor – This is a popular model on Kaggle community. However like MLP regressor, XGBoost regressor took too long to tune.
- Evaluating different models
  - Decision tree regressor started with metrics below the benchmark set however with parameter tuning the model proved to be the best choice. It also took the least time to train.

  - Bagging over decision tree regressor improved the scoring however took a very long training time
  - MLP regressor and XGBoost regressor were quite slow to train and bagging and boosting could not be applied on them
- Compare results on Kaggle
  - I uploaded the results from Decision tree regressor, MLP regressor and XGBoost regressor on Kaggle.
  - The mean score on the competition was 0.892. For the test results evaluated in this project, the scores were as below

Model	RMSLE Score
Kaggle mean score	0.892
Decision tree regressor with bagging	0.448
MLPRegressor	0.480
XGBoostRegressor	0.520

Based on the results displayed in above section and the learnings from this project I conclude the following:

- Comparison Decision tree regressor, MLP regressor and XGBoost regressor is provided below.

- When a complex feature is broken down into multiple simple features, the model improved.
- Ridge regressor produced better results to begin with however the results did not change based on parameter tuning. Hence ridge regressor was selected as benchmark
- Decision tree regressor performed the best with parameter tuning followed by bagging and boosting

- MLP regressor produced good R2 and RMSLE scores however the prediction data was not usable because it MLP regressor had predicted negative time in some cases.

The most interesting aspect of this project was data exploration. The huge dataset only had a limited number of features. I was amazed at the improvement of R2 scores when I derived simple features from complex ones. Another interesting aspect was model tuning where I had to manually use divide and conquer technique to tune the model parameters however this was because of machine configuration limitations. The details about the challenges I faced are provided in section 4c Complications and challenges faced during implementation the final model is good to predict the times however I would have liked to apply bagging and boosting over MLPRegressor and compare the results however my machine configurations do not allow me to do so.

7. Improvements

Few improvements on this project which can be tried and I could not try because of machine configuration limitations are as below:

- Apply Grid Search CV to get the best possible tuning parameters for both DecisionTreeRegressor and MLPRegressor
- Apply bagging and boosting on MLPRegressor

8. Versions of software and libraries used

Following software / libraries will be used for this project

- Jupyter notebook - 3.6
- Python 3.6.2
- Numpy - 1.13.0
- matplotlib - 2.0.2
- pandas - 0.20.2
- sklearn - 0.18.1
- seaborn - 0.7.1

Other utility libraries like math and time will also be used.

9. References

- Kaggle Link: https://www.kaggle.com/c/nyc-taxi-trip-duration
- RMSLE Link: https://www.kaggle.com/c/wikichallenge/details/Evaluation