Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

# New York City Taxi Trip Duration

*Authors:*
Federico De Servi - 812166 - f.deservi1@campus.unimib.it
Federico Luzzi - 816753 - f.luzzi1@campus.unimib.it
Christian Uccheddu - 800428 - c.uccheddu@campus.unimib.it

January 6, 2021

## Abstract

The aim of this project is to predict the duration of a taxi journey in New York city starting from a series of information regarding the geospatial data of departure and arrival, the number of passengers and information on the period in which this journey is made. This work approaches the task by integrating useful external data and applying deep learning techniques to model. The approach and the models applied give good results. In particular one of the models tried reaches satisfying performances with an MSE of 0.2195

Keywords: *Machine Learning, trip duration, Neural Network*

## Contents

## List of Figures

## 1 Introduction

In highly populated cities it is important but difficult to predict the duration of a trip due to traffic; in particular, the aim of this project is to know the duration of a taxi journey in New York starting from a series of information regarding the geospatial data of departure and arrival, the number of passengers and the information on the period in which this journey is made.

One of the possible applications is to help the user predict the duration of his taxi journey in advance when making his reservation.

The approach to this type of problem was to integrate information regarding the area in which the trip takes place as there can be very large variations in time based on the place; to make data enrichment and have more features available on which to train a neural network. The choice to use neural networks was made because there is a very large number of data and these algorithms allow you to easily find non-linear relationships between the input variables

## 2 Datasets

The dataset was taken from kaggle[1]. Let's proceed with the description of the dataset's structure and with the choices made in order to have more useful data.

### 2.1 Time Data

Traffic is highly dependent on the time of day a trip is made. Precisely for this reason the start hour of each trip was extrapolated and a dummy variable that took into account this hour was created.

Also, the week day information (monday, tuesday, wednesday, ...) has been extrapolated because it is thought that the road conditions strongly depend on the day of the week, in particular for working days. Dummy variables have been created to account for the day of the week.

In order to improve the spatial information of latitude and longitude, three new variables that handle better those informations have been defined:

$$x = \cos(\text{lat}) \cdot \cos(\text{lon}) \quad y = \cos(\text{lat}) \cdot \sin(\text{lon}) \quad z = \sin(\text{lat}) \tag{1}$$

The idea is to represent those latitudes and longitudes via 3 coordinates. The problem with latitude and longitude is that they are 2 features representing a 3-dimensional space. Three dimensions can be found using the sine and cosine functions as can be seen in the equation 1. In this way, these features can be normalized properly. In order to do this correctly all the coordinates would be multiplied for the Earth's Radius but, since the data must be scaled, this operation can be omitted[2].

### 2.2 Data enrichment

Given that in a city like New York the area also has a significant impact on traffic, it was decided to add data that, starting from the geospatial coordinates, took into account the county in which that point is located. To do this, the shape files of the counties[3] of New York was taken and through *geopandas* (a python library for handling geospatial data) it was possible to trace the point where each place is located.

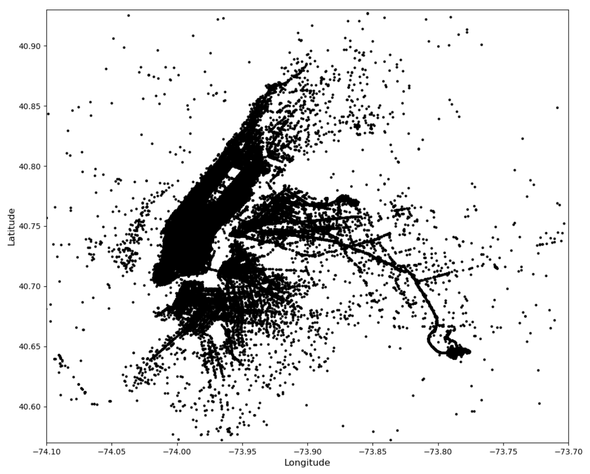


Figure 1: map without county information

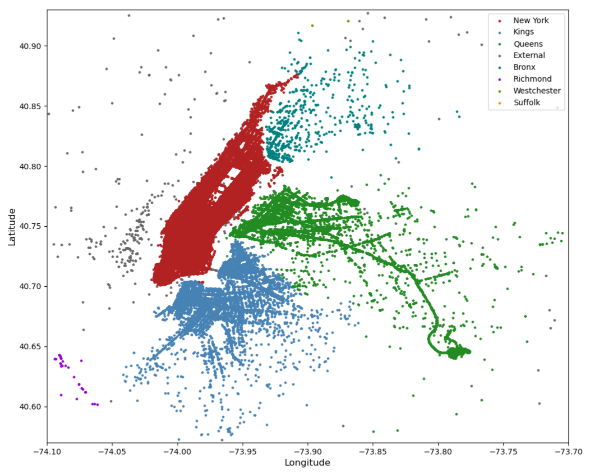


Figure 2: map with county information

As can be seen from the map in the Figure 2, most of the observations lie on the center of New York and only a few observations take place in other county, in particular the number of displacements for each county is:

| County | Number of pickups |
|---|---|
| New York | 911193 |
| Queens | 37196 |
| Kings | 17569 |
| Bronx | 811 |
| External | 529 |
| Richmond | 40 |
| Nassau | 9 |
| Westchester | 7 |
| Suffolk | 4 |

Table 1: Number of pickups for each county

As can be seen in the Table 1 most of the County in which the taxi trip starts is New York.

## 2.3 Manhattan distance

Starting from the geospatial coordinates it was decided to use the Manhattan distance to calculate the distances between the points. This choice is obviously an approximation of the real distance between the points; a more accurate calculation could have been done using Google Maps but this computation was excessively expensive.

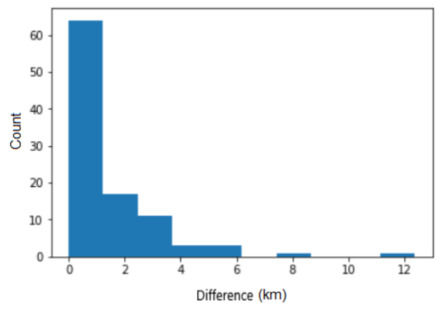


Figure 3: Difference between Manhattan distance and Google Maps distance

As can be seen from Figure 3, 100 random records were taken and both the Manhattan distance and the real distance were calculated through Google Maps; in particular the difference averaged 1.85 km with a standard deviation of 1.43 km centering itself as a Poisson. The Manhattan distance seemed like a good compromise between computational efficiency and real value because it performs very well on small distance.

## 2.4 Splitting

The data was divided into training and validation preserving the distribution of the target variable. The proportion of the train and validation sets is 70% - 30%, this was done to have a number of elements in the validation set comparable to the number of elements of the test set.

## 2.5 Outliers

By plotting the distributions of all the variables it has been realized that there were several outliers. In particular, there were trips lasting few seconds and trips lasting many hours as can be seen in Figure 4 (with destination evidently outside the city). It was therefore decided to delete the last 5% of the data based on the target variable. In addition, all trips with a duration of less than 10 seconds (about 3000 observations) have also been removed as it may be that they are only detection errors. Completely incorrect observations were also removed as they had travel times of less than half an hour for distances greater than 100 km; for this reason the observation with a computed distance more than 80 km have been removed (12 records).

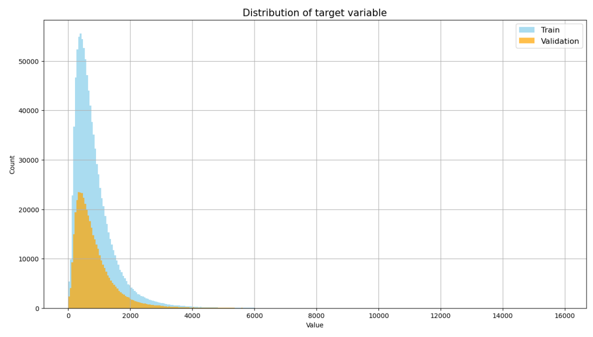


Figure 4: Before preprocessing

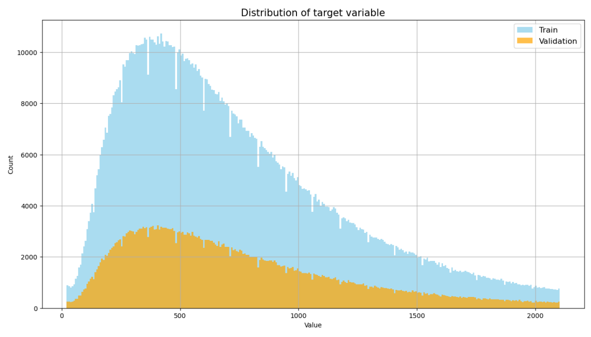


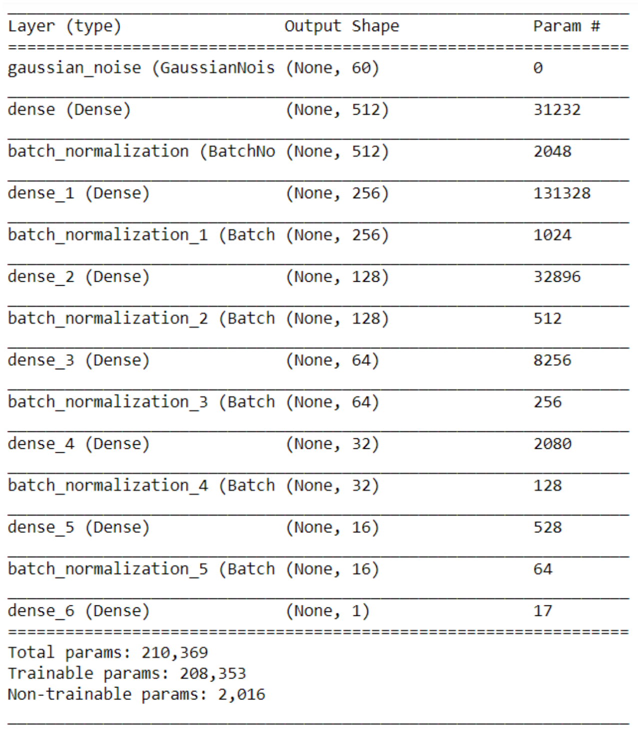
Figure 5: After preprocessing

As can be seen from Figure 4 and Figure 5 there are several observations which are considerably larger than all the others.

## 3 The Methodological Approach

Given the large number of records, it was decided to use neural networks to solve the problem of predicting the taxi trip duration. The choice of the model and the hyperparameters was made after several empirical tests; below there is the structure of the three models with the best performance:

### 3.1 Model comparison



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gaussian_noise (GaussianNois (None, 60)                0
_________________________________________________________________
dense (Dense)                (None, 512)               31232
_________________________________________________________________
batch_normalization (BatchNo (None, 512)               2048
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_3 (Batch (None, 64)                256
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
batch_normalization_4 (Batch (None, 32)                128
_________________________________________________________________
dense_5 (Dense)              (None, 16)                528
_________________________________________________________________
batch_normalization_5 (Batch (None, 16)                64
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 17
=================================================================
Total params: 210,369
Trainable params: 208,353
Non-trainable params: 2,016
_________________________________________________________________
```

Figure 6: Summary model 1

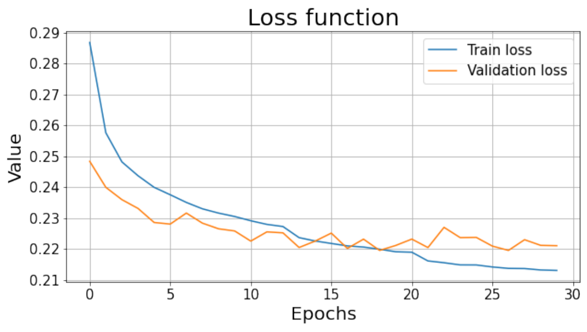


Figure 7: Loss model 1

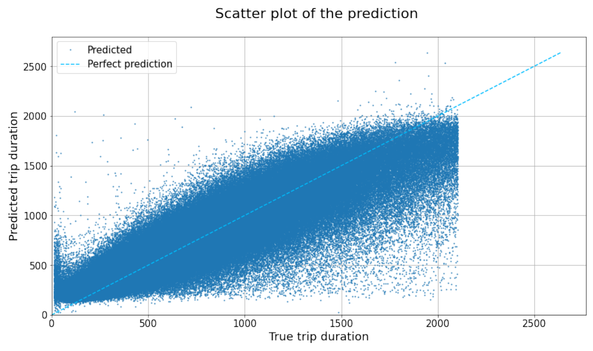
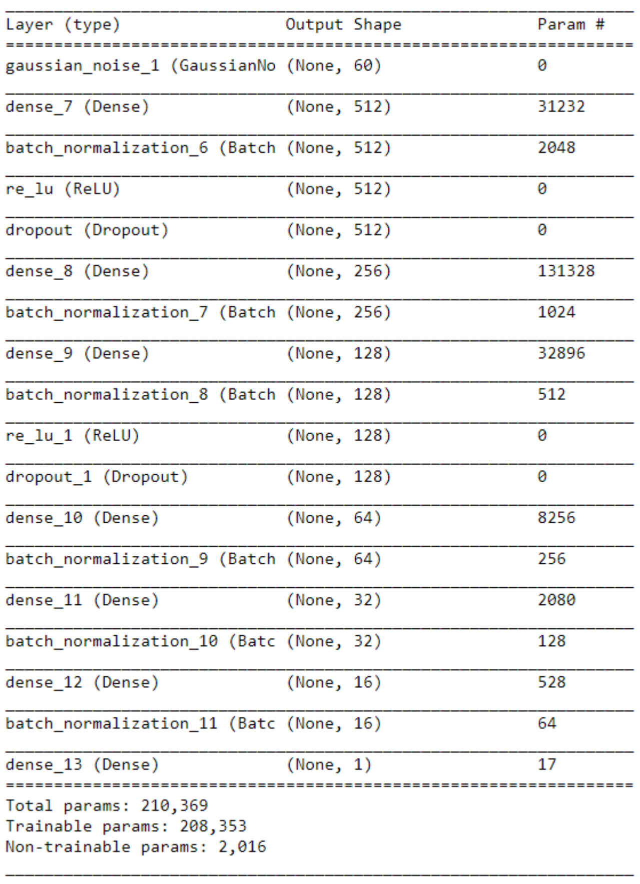


Figure 8: Scatter model 1



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
gaussian_noise_1 (GaussianNo (None, 60)                0         
_________________________________________________________________
dense_7 (Dense)              (None, 512)               31232     
_________________________________________________________________
batch_normalization_6 (Batch (None, 512)               2048      
_________________________________________________________________
re_lu (ReLU)                 (None, 512)               0         
_________________________________________________________________
dropout (Dropout)            (None, 512)               0         
_________________________________________________________________
dense_8 (Dense)              (None, 256)               131328    
_________________________________________________________________
batch_normalization_7 (Batch (None, 256)               1024      
_________________________________________________________________
dense_9 (Dense)              (None, 128)               32896     
_________________________________________________________________
batch_normalization_8 (Batch (None, 128)               512       
_________________________________________________________________
re_lu_1 (ReLU)               (None, 128)               0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_10 (Dense)             (None, 64)                8256      
_________________________________________________________________
batch_normalization_9 (Batch (None, 64)                256       
_________________________________________________________________
dense_11 (Dense)             (None, 32)                2080      
_________________________________________________________________
batch_normalization_10 (Batc (None, 32)                128       
_________________________________________________________________
dense_12 (Dense)             (None, 16)                528       
_________________________________________________________________
batch_normalization_11 (Batc (None, 16)                64        
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 17        
=================================================================
Total params: 210,369
Trainable params: 208,353
Non-trainable params: 2,016
_________________________________________________________________
```

Figure 9: Summary model 2

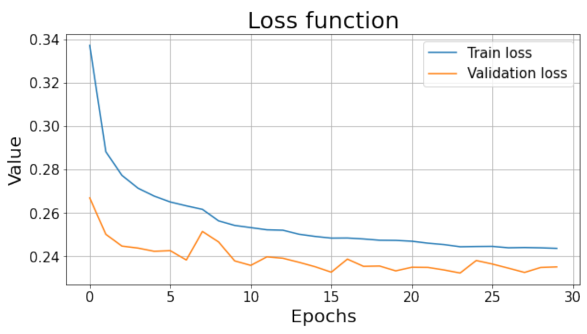


Figure 10: Loss model 2

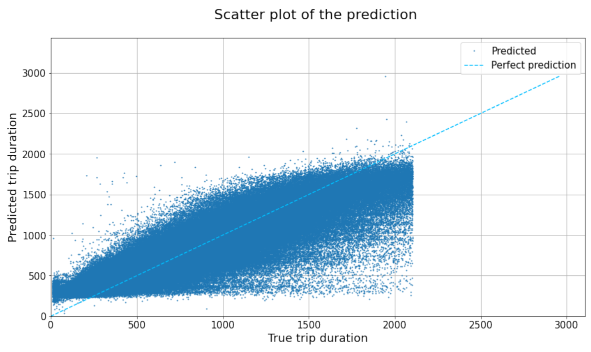
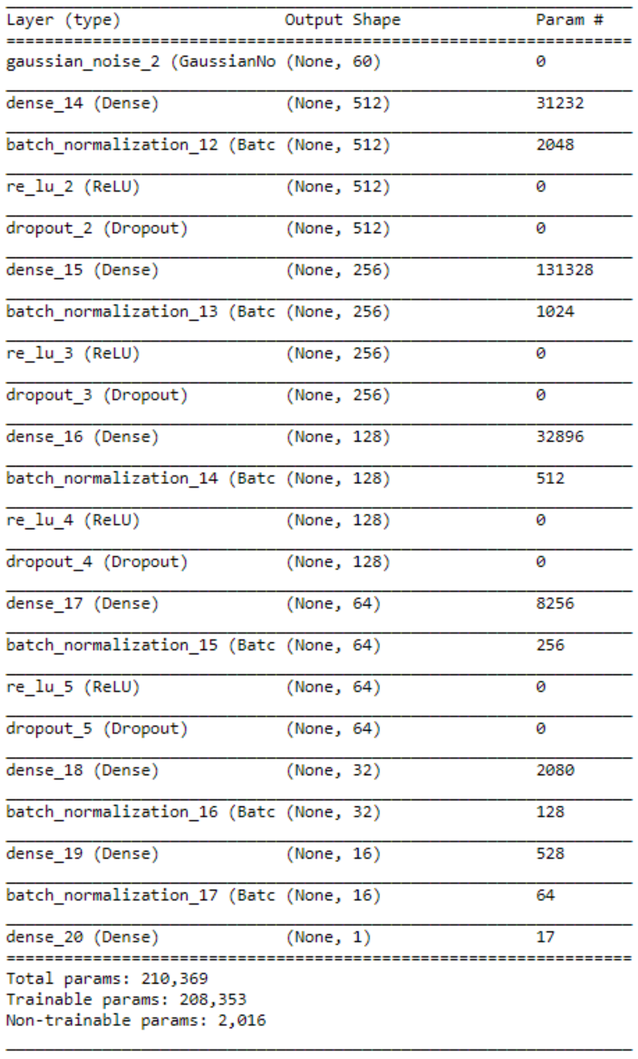


Figure 11: Scatter model 2



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
gaussian_noise_2 (GaussianNo (None, 60)                0         
_________________________________________________________________
dense_14 (Dense)             (None, 512)               31232     
_________________________________________________________________
batch_normalization_12 (Batc (None, 512)               2048      
_________________________________________________________________
re_lu_2 (ReLU)               (None, 512)               0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_15 (Dense)             (None, 256)               131328    
_________________________________________________________________
batch_normalization_13 (Batc (None, 256)               1024      
_________________________________________________________________
re_lu_3 (ReLU)               (None, 256)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_16 (Dense)             (None, 128)               32896     
_________________________________________________________________
batch_normalization_14 (Batc (None, 128)               512       
_________________________________________________________________
re_lu_4 (ReLU)               (None, 128)               0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_17 (Dense)             (None, 64)                8256      
_________________________________________________________________
batch_normalization_15 (Batc (None, 64)                256       
_________________________________________________________________
re_lu_5 (ReLU)               (None, 64)                0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_18 (Dense)             (None, 32)                2080      
_________________________________________________________________
batch_normalization_16 (Batc (None, 32)                128       
_________________________________________________________________
dense_19 (Dense)             (None, 16)                528       
_________________________________________________________________
batch_normalization_17 (Batc (None, 16)                64        
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 17        
=================================================================
Total params: 210,369
Trainable params: 208,353
Non-trainable params: 2,016
_________________________________________________________________
```

Figure 12: Summary model 3

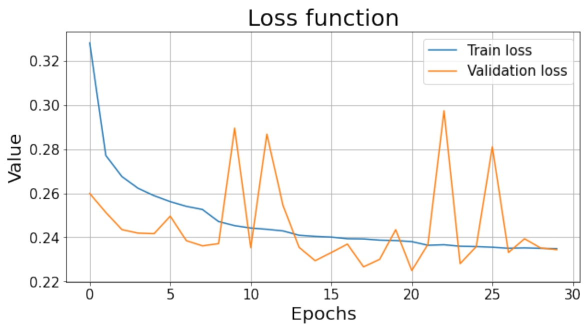


Figure 13: Loss model 3

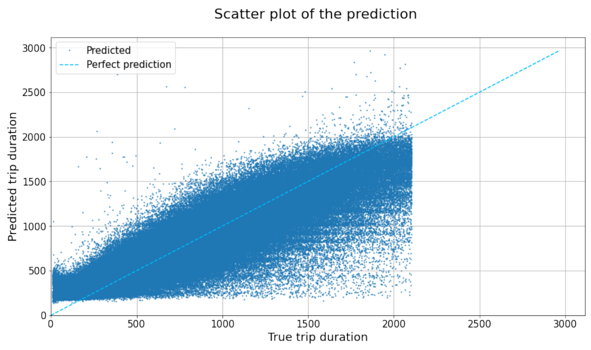


Figure 14: Scatter model 3

### 3.2 Hyperparameters optimization

After choosing the model with the best performance (in this case model 3) we moved on to optimizing the hyperparameters for this model; in particular a detailed description of the hyperparameters is left below:

- Learning rate: One of the biggest problems was that the excellent premises were very tight, the choice of an adequate learning rate was therefore fundamental. It was therefore decided to put an adaptive learning rate, which would change going forward with the epochs in order to be able to quickly find an optimum point and then move carefully in the vicinity of this point so as not to skip it. In particular the scheduler for the learning rate is the following:

  ```
  def lr_scheduler(epoch, lr):
      if epoch <= epoch_1:
          lr = lr_1
      if (epoch > epoch_1) & (epoch < epoch_2):
          lr = lr_2
      if epoch > epoch_2:
          lr = lr_3
      return lr
  ```

  This scheduler was manually made after several observations on the specific model, in order to improve its training.

- Dropout: It was included because empirically it was found that it helped prevent overfit problems. The dropout values change for each layer of the network and have been chosen empirically.
- Batch Size: The batch size have been set to 128. It was done because it is a good compromise between speed and number of iterations.
- Epochs: It was decided to choose 30 epochs because we realized that it is a good compromise between execution times and generalization error. However, an earlystopping callback has been added to make sure you don't have too many epochs in which the model's loss is actually not improving. In addition, a checkpointer callback has been defined in order to save only the weights of the model that reached the best validation loss between all epochs.

```
auto_save = ModelCheckpoint(args.output +"/current_model",
                            monitor='val_loss',
                            save_best_only=True,
                            save_weights_only=True,
                            mode='auto')
```

- Gaussian Noise: Apply additive zero-centered Gaussian noise with a standard deviation of 0.005. This parameter is useful to mitigate overfitting. Gaussian Noise (GS) is a natural choice as corruption process for real valued inputs.
- Batch Normalization: Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). [4] The parameters used in this layer are the default parameters of keras.

### 3.3 Evaluation measures

Quantitative measures were used to evaluate the different models in addition to the observation of the scatter plot. In particular, MSE has been used:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\text{true}_i - \text{pred}_i)^2 \tag{2}$$

Evaluating the MSE it can be said that the best model is model number 1. The next section presents the graphs related to the best model.

## 4 Results and Evaluation

Based on the following MSE measures, the best model is **Model1**

| **Model1** | Model2 | Model3 |
|---|---|---|
| **0.2195** | 0.2323 | 0.2249 |

Table 2: MSE on validation

Below are the results of the model with the best performance

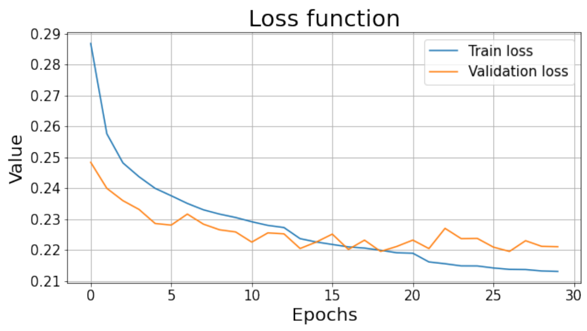


Figure 15: Loss best model

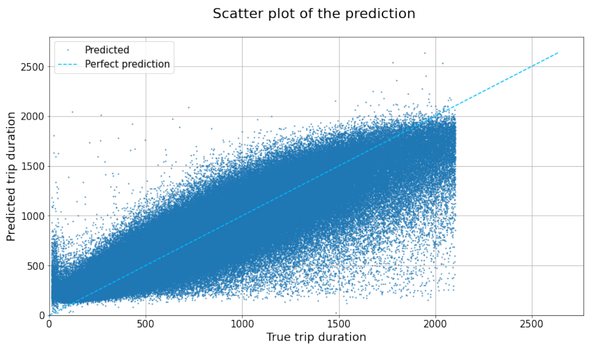


Figure 16: Scatter best model

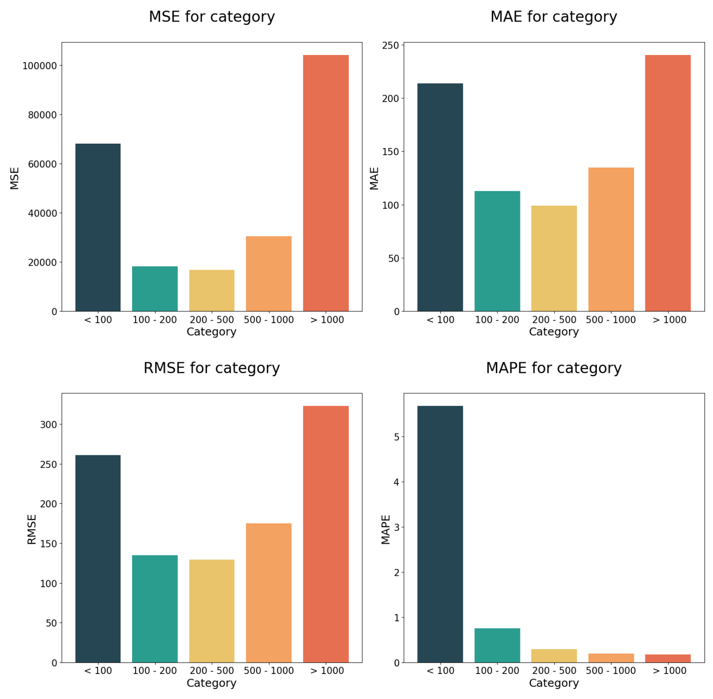


Figure 17: Error for different classes

# 5 Discussion

As can be seen in Figure 17 the data have been splitted into five different categories in order to see the errors among this categories. In particular:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\text{true}_i - \text{pred}_i| \tag{3}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{true}_i - \text{pred}_i)^2} \tag{4}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\text{true}_i - \text{pred}_i|}{\text{true}_i} \tag{5}$$

have been defined. As can be seen from the equation 5 the category which is best predicted is the one with the highest trip duration, whereas the shortest trip duration have been predicted badly.

## 5.1 Future improvements

There are several things that could be improved:

- Information about the average speed at which the movement occurs could be added.
- Information about the angle at which the move is made could be added to take into account how far you actually deviate from the Manhattan distance.
- The Google Maps-like distance between points could be used.
- Information about traffic during the trip.

Future improvements for this project could try to use the data that has been cut previously (even if it represents a small a percentage of the total) and try to apply models that could handle those parts of the original dataset.

## 6 Conclusions

The approach applied gives good results. In particular, the integration with geospatial data is important in order to consider different types of trips and also the outlier cut of too long trips and too short to be meaningful.

The models tried are quite complex but the best is the simplest one (5 layers instead of 7 layers), usually a simpler model avoids overfitting and is more robust.

The project could be improved in the future by adding some traffic information or based on this model bring a real-time one, that can be much more useful.

## References

[1] MultiMedia LLC. MS Windows NT kernel description, 1999.

[2] Younes Charfaoui. Working with geospatial data in machine learning, 2019.

[3] Kaggle. Neighborhoods in new york, 2017.

[4] Towards Data Science. Batch normalization in neural networks, 2017.