MACHINE LEARNING FINAL PROJECT

BY

CHRIS LEE, WILLIAM SALAS & JUAN TORDOYA

MAY 2020

MSA 8150

Abstract

Organizations and businesses are currently more than ever highly motivated turning to predictive analytics to help solve their demanding problems and to discover new opportunities they could not have foreseen. Not too long ago, if we were to give a computer an input of images and asked the computer to tell us what is in the image, the computer could not comprehend. However, it is now showing up in all sorts of different kinds of software applications. Our motivation spiked our interest to learn more about and how to use neural networks to build a customized image recognition system for our first chosen project, Lung Xray. With these ideas in mind, we were also motivated to choose for our second project to be Gotham City Cabs.

Large - Project: X-Ray:

In this project we are customizing a neural network to perform the task of passing a set of images of lungs to identify one of three types of lung problems that pose serious threats to an individual's respiratory system.

Small - Project: Gotham City Cabs:

Given a relatively large amount of recorded taxi durations from various parts of Gotham. We use this dataset to gather a better understanding of taxicab duration time to increase the efficiency of Batman's dispatch and rescue time.

Conclusion:

It was reassuring to develop numerous amounts of different neural networks and models that took in multiple different features to get increasing accuracy of predictions of our projects. We have customized a neural network in classifying the different types of lung diseases given an x-ray image of a lung. We have also predicted the durations of a taxi-cab ride in the city of Gotham. In our models we found that random forest and boosting to be our most efficient models. However, in our neural networks and models, to better our model's performances, we could have implemented more tuning and neural network feature interactions.

Chapter 1: Large - Project; Lung X-Ray

Introduction:

With the advances in machine learning techniques, we are presented with the task to classify x-ray images from a dataset of 2393 images representing three different types of lung illnesses. We will fit a classification model using the given data, and then predict the class of test images

Problem analysis:

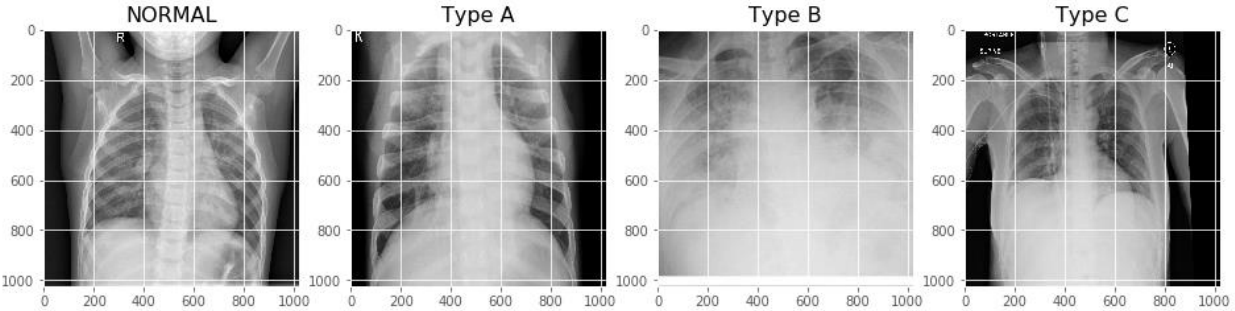
The increase in the number of COVID-19 cases presents a problem for medical diagnosis doctors that need to evaluate several thousands of x-ray images, making the process prone to human error. Our team has been tasked with creating a machine learning model using convolutional networks (CNN) for image classification, to reduce the human error, and produce a preliminary classification of lung x-ray, making the evaluation and diagnosis task faster and more accurate.

This process is not meant to replace the medical professional diagnosis, but to aid in the classification of possible diagnostics

Datasets:

We are working with 2393 1024x1024 x-ray images, labeled correctly within four different diagnosis groups as follows:

- 1212 images without detected issue, classified as ***Normal***.
- 643 images diagnosed with disease A, named ***Type-A***.
- 253 images diagnosed with disease B, named ***Type-B***.
- 285 images diagnosed with disease C, named ***Type-C***.



Data Processing:

Due to the fact that we are working with 2389 images, each one of them represented by a 3x1024x1024 matrix, with values between 0 and 255, our model is going to process 2389x3x1024x1024 values of type *float*. Then, the pre-process is more oriented to avoid memory issues, getting a time reduction during the model fitting.

Every image is converted into a grayscale image to reduce the dimensions from 3x1024x1024 to 1024x1024, followed by standardization, which is just a scale change in the matrix representation values, for us to work with values between 0 and 1, instead of values between 0 and 255. Also, as part of the memory optimization, each value of the matrix is stored as *float32*, gaining memory capacity without sacrificing much numerical precision. Reducing from 17 digits to 9 is enough for usual tolerances used in optimization problems.

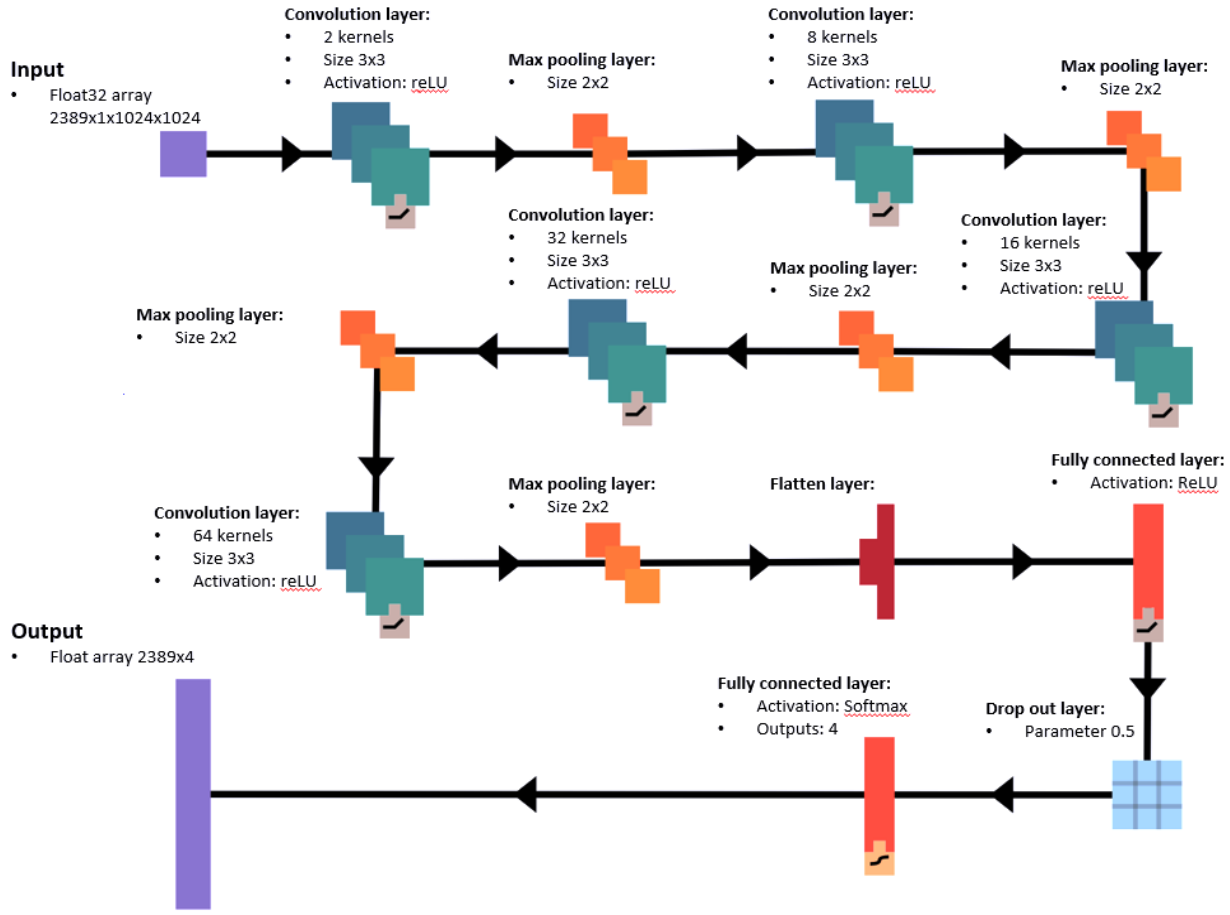
Model Selection:

The most used model for disease diagnostic is based on convolutional neural networks, that's why we fitted some different models, varying the layers of the neural network, the number and size of kernels for those convolutions, the pooling size and the kind of pooling following each convolution.

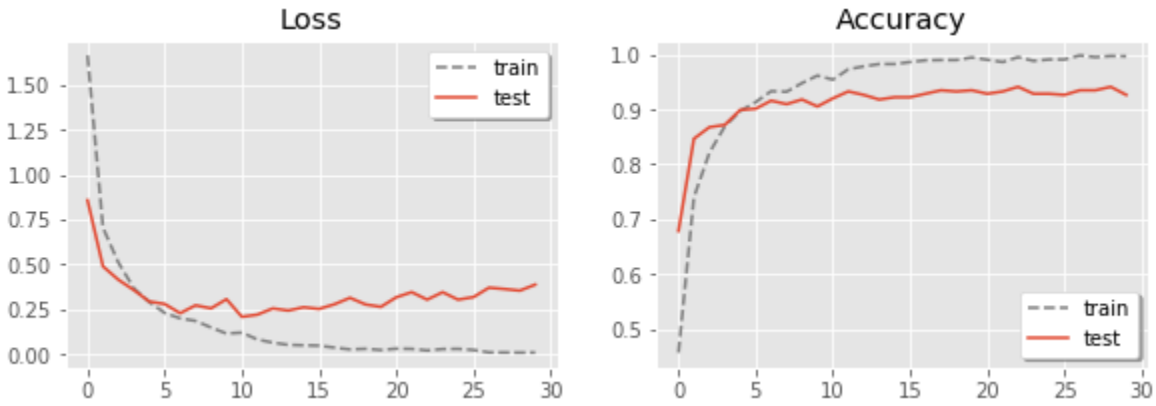
It takes a long time to fit a convolutional neural network with our kind of data, due to that limitation, we selected the best model based on a train/test performance comparison, using a configuration 80/20. The results are summarized as follows:

Model	Model 1	Model 2	Model 3	Model 4	Model 5
Number of filters configuration	[2,4,8,32,64]	[2,4,8,32,64]	[2,8,8,32,64]	[2,8,16,32,64]	[2,8,16,32,64]
Kernels size configuration	[2,2,2,2,2]	[2,2,2,2,2]	[2,2,2,2,2]	[2,2,2,2,2]	[3,3,3,3,3]
Kind of pooling	Max	Average	Max	Max	Max
Pooling size dimensions	[2,2,2,2,2]	[2,2,2,2,2]	[2,2,2,2,2]	[2,2,2,2,2]	[2,2,2,2,2]
Train accuracy Test accuracy	0.9375 0.9160	0.8829 0.8929	0.9291 0.8971	0.9333 0.9223	0.9438 0.9286

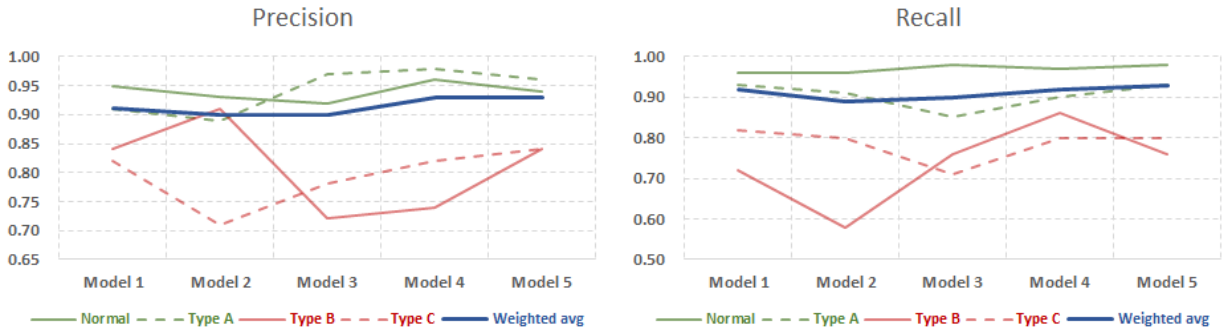
To explain briefly each model structure, model 1 has five double layers, each one composed by one convolution and one pooling, the double layer 3, for example, is explained by the third component of each configuration in the table, based on 8 kernels convolution, each kernel with size 2x2, followed by a 2x2 max pooling. For each model tested, the last convolution-pooling layer is followed by a flatten layer, a fully connected layer with activation function ReLU, a drop out layer with parameter 0.5, and a last fully connected layer of 4 outputs and activation 'softmax' activation function. In addition, each model was fitted using 10 epochs and 256 batch size. Finally, the models we described look as follows:

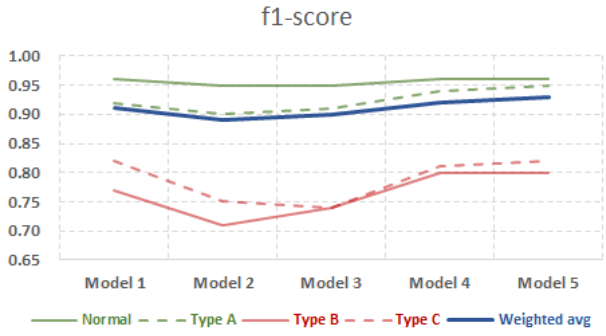


We decided to compare models at 10 epochs, due to after that number of epochs, we observed a generalized overfitting behavior. The next graph shows the exposed situation for model 5.



To consider more factors at the moment of choosing the best model, we compared the precision, recall, f1-score and the respective weighted average values for each measure, and each model. As a result, we decided the model 5 is the one with best performance to diagnose the condition of each patient, due the loss of a little piece of precision for Normal and Type A conditions, to gain precision and recall for Types B and C, with less support in the sample we are working with.





Results:

Based on the performance of each model, we have selected Model 5 as our most accurate and efficient model. The model metrics shown below

	Precision	Recall	f1-score	Support
Normal	94%	98%	96%	245
Type A	96%	93%	95%	136
Type B	84%	76%	80%	50
Type C	84%	80%	82%	45
Macro Avg.	90%	87%	88%	476
Weighted Avg.	93%	93%	93%	476

Actual \ Pred	Normal	Type A	Type B	Type C	Total
Normal	241	1	2	1	245
Type A	9	127	0	0	136
Type B	4	2	38	6	50
Type C	2	2	5	36	45
Total	256	132	45	43	476

Conclusion:

We opted to work with convolutional networks for image classification for several reasons, mainly because of the dimensionality reduction, which fits with the high number of parameters in our images. In a neural network, the number of parameters grows with the number of layers, making the task of image analysis very computationally hard. Due to the dimensional reduction algorithms, the training and computational effort is diminished with convolutional networks.

Additionally, CNNs are fully connected feedforward neural networks, which means the connections between nodes do not form a cycle. In these networks, the information moves in one direction from the input through the hidden nodes to the output with no cycles or loops. Therefore, these networks are very effective at reducing the number of parameters while preserving the original quality of the image.

We design our CNN with an incremental number of filters, which means the firsts layers will learn fewer filters than the final layers. This architecture is a common practice to design CNNs, so we followed this practice by creating five layers of 2,8,16,32 and 64 respectively. Using the powers of 2 for each value is also a common practice. The max pooling for each layer was set to (2,2) to reduce the spatial dimensions. Our Kernel size was set to 3X3, we learned that this kernel size aids in learning larger spatial filters while reducing the volume of the image.

The final steps of our CNN after flattening the values, we go through two fully connected layers, the first dense layer has an output of 256 units and a dropout rate of 0.5 to avoid overfitting. The last layer's output is the same size as the number of classes (4).

In conclusion, training CNNs is a tedious trial and error process, with multiple iterations until we are able to find the most accurate model. We did not have unlimited time and computer power to try more sophisticated networks, however, we were able to achieve encouraging results with our model. We would most likely continue our research for better models, and additional data. One key element that we notice is that we could supplement the image data with additional metadata of the image, that would give us additional features to predict the illness.

Chapter 2: Small - Project; Gotham

Introduction:

In order to predict taxi trip time while Batman's Batmobile is broken, we have to come up with a prediction model that fits our needs. We trained several different models until getting the one with the highest accuracy rate.

It is obvious that the shortest distance between the two points is not the best indicator of actual distance, because that distance would give the distance in the plane, not accounting for the roads and traffic. However, since we don't have a better way to know the city's layout, and the traffic conditions.

We will use past data to try to predict future durations. The process will take the following steps:

1. Data Processing. Including data cleaning, splitting, and calculations
2. Training testing split
3. Model selection.
4. Test the model with testing data

Data Processing & Problem Analysis:

We started with a simple data structure:

Pickup_datetime: datetime variable of pick up. We will split this variable to extract individual values for our model. We will try different selections: Hour, minutes, day of week, etc.

NumberOfPassagers: We have to assume that Batman is one of the passengers, and that he will may or may not have to make several trips to save multiple people.

Pickup_x, Pickup_y : Coordinates of pick up location

Dropoff_x, Dropoff_y: Coordinates of dropoff location

Duration: Is the variable we would like to predict

We divided the variables and calculated values in numerical and categorical variables.

1. **Numerical Variable(s):** The numerical variables used are: pickup_x, pickup_y, dropoff_x, dropoff_y, Distance, and CDistance.

a. **pickup_x:** An original variable that represents the x coordinate where the passenger(s) was picked up.

b. **pickup_y:** An original variable that represents the y coordinate where the passenger(s) was picked up.

c. **dropoff_x:** An original variable that represents the x coordinate where the passenger(s) was dropped up.

d. **dropoff_y:** An original variable that represents the x coordinate where the passenger(s) was dropped up.

e. **Distance:** Distance is a new variable calculated using the distance formula. This is assuming Batman travels in a straight line. (Normalized Distance)

$$\sqrt{(dropoffx - pickupx)^2 + (dropoffy - pickupy)^2}$$

f. **CDistance:** CDistance is a new variable with the condition that is a limit on the number of passengers that Batman transports. (Normalized CDistance).

Distance *If NumberOfPassager<=4*

Distance/3 *if NumberOfPassagers>4*

2. **Categorical Variable(s):** The categorical variables used are: (Seconds, Minute, MinuteLevel, Hour, DayOfWeek, Month, and Year) derived from the original variable pickup_datetime, and NumberOfPassengers.

    a. **pickup_datetime:** An original variable that indicates the date and time the passenger(s) was picked up. This variable will be split into multiple new variables.

        i. **Year:** Year the passenger(s) was picked up. This variable was dropped due to all of the observations being in the same year (2034).

        ii. **Month:** Month the passenger(s) was picked up. This variable was converted into a categorical variable since all the numerical values under this label range from 1 to 7.

        iii. **Day:** Day the passenger(s) was picked up. This variable was deemed unnecessary and dropped due to the following variable, DayOfWeek.

        iv. **DayOfWeek:** Day of the week that the passenger(s) was picked up; for example, 1 represents Monday, 2 represents Tuesday, …, 7 represents Sunday.

        v. **Hour (Numerical):** Hour the passenger(s) was picked up.

vi. **Minute:** Minute the passenger(s) was picked up. This variable is Numerical (ranging from 0 to 59) but was converted to Categorical.

1. **MinuteLevel:** Splitting the numerical variable into two levels, where the first 30 minutes, 0 - 29 will be level 1, and the second 30 minutes, 30 - 59 will be level 2.

vii. **Second:** Second the passenger(s) was picked up. This variable was dropped for being highly insignificant in affecting the response variable and because the testing_file has no values for seconds, all '00'.

b. **NumberOfPassengers:** The number of passengers that were being transported by the cab. This variable will be used for the condition that there was a max limit on how many passengers can be picked up per trip. Assuming Batman maximizes his trips by shoving as many passengers as he can see per trip, or limiting his cab by a total of 5 passengers.

3. **Response Variable(s):** The response variable is: Duration.

a. **Duration:** Duration is the duration of the trip from pickup to drop off in seconds. This is the variable we will try to predict.

Datasets:

After applying the data transformations, we created three main datasets that will be used to train our model and to be compared to see which performs the best.

1. **Original:**

NumberOfPassengers	duration	pickup_x	pickup_y	dropoff_x	dropoff_y	Month	DayOfWeek	Hour	Minute
1	434	135.4281135	321.1907733	141.3177116	321.3578023	6	4	20	47
1	415	148.7812906	322.5655359	157.145868	337.0526545	4	6	8	0
2	398	124.0000056	313.3477631	131.362606	300.6135847	4	5	18	57
1	746	124.2884126	316.4954039	143.2843505	334.5421161	4	6	22	40
1	102	177.3448612	358.0878439	168.7094893	363.038196	5	1	11	39

'data.frame': 1000000 obs. of 10 variables:

$ NumberOfPassengers: Factor w/ 9 levels "0","1","2","3",..: 2 2 3 2 2 2 3 3 2 2 ...

$ duration : int 434 415 398 746 102 445 234 673 241 428 ...

$ pickup_x : num 135 149 124 124 177 ...

$ pickup_y : num 321 323 313 316 358 ...

$ dropoff_x : num 141 157 131 143 169 ...

$ dropoff_y : num 321 337 301 335 363 ...

$ Month : Factor w/ 7 levels "1","2","3","4",..: 6 4 4 4 5 5 6 6 5 5 ...

$ DayOfWeek : Factor w/ 7 levels "1","2","3","4",..: 4 6 5 6 1 6 4 6 1 2 ...

$ Hour : Factor w/ 24 levels "0","1","2","3",..: 21 9 19 23 12 9 20 20 11 21 ...

$ Minute : int 47 0 57 40 39 13 32 41 32 18 ...

2. Distance:

NumberOfPassengers	duration	pickup_x	pickup_y	dropoff_x	dropoff_y	Month	DayOfWeek	Hour	Distance	MinuteLevel
8	87	170.856163	330.3195502	153.4379219	269.9102056	1	6	1	62.87	1
7	12	139.7336884	266.2354305	138.9513072	265.8164939	5	1	22	0.89	2
7	21	29.45094932	663.0197866	28.67910238	662.6053216	5	7	19	0.88	2
6	1524	141.6005495	330.549182	189.8964808	386.1713323	4	3	11	73.66	1
6	808	147.1129175	307.8584622	127.9501467	306.583085	6	7	22	19.21	2
6	991	126.8567944	286.6620747	128.813978	316.0585705	5	1	15	29.46	2
6	492	156.5386306	346.8276593	159.354873	371.0273734	7	5	18	24.36	1
6	563	165.5443457	361.880967	166.7380953	339.9556671	4	5	8	21.96	2

'data.frame': 1000000 obs. of 11 variables:

$ NumberOfPassengers: Factor w/ 9 levels "0","1","2","3",..: 2 2 3 2 2 2 3 3 2 2 ...

$ duration : int 434 415 398 746 102 445 234 673 241 428 ...

$ pickup_x : num 135 149 124 124 177 ...

$ pickup_y : num 321 323 313 316 358 ...

$ dropoff_x : num 141 157 131 143 169 ...

$ dropoff_y : num 321 337 301 335 363 ...

$ Month : Factor w/ 7 levels "1","2","3","4",..: 6 4 4 4 5 5 6 6 5 5 ...

$ DayOfWeek : Factor w/ 7 levels "1","2","3","4",..: 4 6 5 6 1 6 4 6 1 2 ...

$ Hour : Factor w/ 24 levels "0","1","2","3",..: 21 9 19 23 12 9 20 20 11 21 ...

$ Distance : num 5.89 16.73 14.71 26.2 9.95 ...

$ MinuteLevel : Factor w/ 2 levels "1","2": 2 1 2 2 2 1 2 2 2 1 ...

3. Conditional Distance:

NumberOfPassengers	duration	pickup_x	pickup_y	dropoff_x	dropoff_y	Month	DayOfWeek	Hour	MinuteLevel	CDistance
8	87	170.856163	330.3195502	153.4379219	269.9102056	1	6	1	1	20.96
7	12	139.7336884	266.2354305	138.9513072	265.8164939	5	1	22	2	0.3
7	21	29.45094932	663.0197866	28.67910238	662.6053216	5	7	19	2	0.29
6	1524	141.6005495	330.549182	189.8964808	386.1713323	4	3	11	1	24.55
6	808	147.1129175	307.8584622	127.9501467	306.583085	6	7	22	2	6.4
6	991	126.8567944	286.6620747	128.813978	316.0585705	5	1	15	2	9.82
6	492	156.5386306	346.8276593	159.354873	371.0273734	7	5	18	1	8.12
6	563	165.5443457	361.880967	166.7380953	339.9556671	4	5	8	2	7.32

'data.frame': 1000000 obs. of 11 variables:

$ NumberOfPassengers: Factor w/ 9 levels "0","1","2","3",..: 2 2 3 2 2 2 3 3 2 2 ...

$ duration : int 434 415 398 746 102 445 234 673 241 428 ...

$ pickup_x : num 135 149 124 124 177 ...

$ pickup_y : num 321 323 313 316 358 ...

$ dropoff_x : num 141 157 131 143 169 ...

$ dropoff_y : num 321 337 301 335 363 ...

$ Month : Factor w/ 7 levels "1","2","3","4",..: 6 4 4 4 5 5 6 6 5 5 ...

$ DayOfWeek : Factor w/ 7 levels "1","2","3","4",..: 4 6 5 6 1 6 4 6 1 2 ...

$ Hour : Factor w/ 24 levels "0","1","2","3",..: 21 9 19 23 12 9 20 20 11 21 ...

$ MinuteLevel : Factor w/ 2 levels "1","2": 2 1 2 2 2 1 2 2 2 1 ...

$ CDistance : num 5.89 16.73 14.71 26.2 9.95 ...

Sensitivity of Response Variables to the input features on our best dataset (Distance):

```
Variable Importances:
             variable relative_importance scaled_importance percentage
1            Distance 255900303360.000000          1.000000   0.777636
2                Hour  29791078400.000000          0.116417   0.090530
3           DayOfWeek  11050808320.000000          0.043184   0.033581
4           dropoff_y   8027473920.000000          0.031370   0.024394
5           dropoff_x   7951055360.000000          0.031071   0.024162
6            pickup_x   6371163648.000000          0.024897   0.019361
7            pickup_y   5198568960.000000          0.020315   0.015798
8               Month   3974552832.000000          0.015532   0.012078
9  NumberOfPassengers    539009024.000000          0.002106   0.001638
10        MinuteLevel    270534176.000000          0.001057   0.000822
```

Model Selection:

Our models are modeled primarily after the three main datasets. As noted above, Dataset 1 uses only the feature selection of the original dataset with no additional inclusions. Dataset 2 implements a new variable, distance, in which we obtained through the distance formula. Lastly, Dataset 3 includes a condition on the new variable, distance, in which there is a max amount of passengers that Batman can transport at once. With Dataset 2 and Dataset 3, the new variables Distance, and CDistance, representing conditional distance, we also normalized them to see if having a normalized distance would make a difference.

Below is a table of details of our models with their Mean Square Error values:

Dataset	Linear Regression	Decision Tree	Boosting	H2O Cluster Boosting
Dataset 1	229752.82	180053.36	88953.97	X
Dataset 2	109518.20	114665.11	70281.29	50788.53
Dataset 2 (Normalized)	109516.39	114696.09	X	X
Dataset 3	115048.97	122433.05	69840.49	54861.62
Dataset 3 (Normalized)	115052.6	122455.46	X	X

Model Analysis:

We first started off with the linear regression model on our datasets to set a standard. From there, we noticed that the dataset without the inclusion of the new variable, distance/cdistance performed terribly in comparison with the datasets that included the new variable. Dataset 1 had a MSE of 229752.82, while all the other Datasets had a MSE about half of that value. From this, we see that the inclusion of the new variable is extremely helpful.

To confirm, we then applied a decision tree model with a default complexity parameter of 0.01. With this in mind, our theory still stands strong. Dataset 1 had a MSE of 180053.36, and again the remaining datasets had a MSE about half that value. From this point, we also noticed that Dataset 3 and Dataset 3 perform very similar to one another; meaning whether we included a condition on distance and passengers made a significant difference. We also noticed that whether we normalized the distance, the performance was not changing either, so from this point, we discarded the normalized distances.

Following the decision tree model, we applied a Boosting model with 1000 trees, maximum depth of 11, and a learning rate of 0.01, which performed significantly better than the previous two models. Dataset 1 now has an MSE of 88953.97. However, with the newly improved model selection, Dataset 2 and 3 still outperformed Dataset 1 with a MSE of around 70000.

For our final model, because boosting is computationally expensive, we moved our model into a cluster to save a lot of memory and time. We also decided to drop Dataset 1 at this point since the introduction of the new variable has been performing better. In the cluster, we tuned our previous boosting method; we doubled the number of trees and adjusted the learning rate and depth to 0.05 and 10, respectively. With this, the mean square error for dataset 2 and dataset 3 are 51831.69 and 54861.62, respectively.

Conclusion:

In conclusion, introducing the new variable, Distance, was assuring. It was able to help generate better models. In addition, whether we had a condition on Distance and whether we normalized the distances, the performance of our models remained intact and performed equally well. Overall, our best model was the model that included the Boosting algorithm of increased trees and learning rate with the introduction of the new variable, Distance, yielding a mean square error of 50788.53.