VEHICLE FORENSICS:
APPLIED MACHINE LEARNING USING VEHICLE DATA

Andrew Ayers

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Congdon School of Supply Chain, Business Analytics, and Information Systems

University of North Carolina Wilmington

2023

Approved by

Advisory Committee

Geoff Stoker	Gulustan Dogan

Ron Vetter, Chair

TABLE OF CONTENTS

Page
Title Page i
Table of Contents ii
Abstract iii
Chapter 1: Introduction 1
Chapter 2: Background Research 2
2.1 Vehicle Forensics 2
2.2 Vehicle Forensics Tools 3
2.3 Machine Learning Models 3
2.3.1 Linear Regression 4
2.3.2 Elastic Net 4
2.3.3 Ridge 4
2.3.4 Lasso 4
2.3.5 ExtraTreeRegressor 5
2.3.6 GradientBoostingRegressor 5
2.3.7 XGBRegressor 6
2.3.8 LGBMRegressor 6
2.4 Related Literature 3
Chapter 3: Proposed Problem 6
3.1 Initial Ideation 6
3.2 Vehicle Data Unavailability 7
3.3 Where to Find Data 8
Chapter 4: Methodology 10
4.1 Data Collection 10
4.2 Data Sanitization 11
4.3 Exploratory Analysis 11
4.4 Machine Learning Model Training 12
4.5 Application of Models 13
Chapter 5: Results 13
5.1 Expected Results 13
5.2 Actual Results 14
5.3 Assessment of Results 16
Chapter 6: Conclusion 16
6.1 Issues 16
6.2 Generated Value 17
6.3 Reflection 17
6.4 Future Works 18
References 19
Appendixes
A. Example trip file download from Autopi……..…….21
B. Autopi CM4 Telemetric Device……..……..…….21
C. Postman GET request for speed data……..……..…….23
D. Postman GET request for Location data.……..…….23
E. Project code for vehicle dataset……..……..…….24
F. Project code for training the models…..……..…….25
G. Project code for applying the models…..……..…….26

Tables

1. Table with field names, data type and description..... 9
2. Table showing the Pandas corr method run on our data……..…….14
3. Table displaying each model with Coefficient Values……..……17

Figures

1. Example of trip data from the Autopi cloud…….....……..8
2. Workflow describing the data cleaning process ……..…….……12
3. The Describe() method run on our dataset ……14
4. Summation of trained models' estimations……...……18
5. Line plot of the first 500 records and model estimations ……..……19

ABSTRACT

Vehicle Forensics: Applied Machine Learning Using Vehicle Data. Ayers, Andrew 2023. Capstone Paper, University of North Carolina Wilmington.

Within the last two decades recent technological advancements surrounding automobiles have changed the landscape and caused vehicles to become huge repositories and generators of digital information. The generated data has many values within both digital forensics and related research fields. Though the data has value, there are not very many openly available methods for accessing this data. In this paper, a dataset was created which was then used in a machine learning algorithm to help assess the data collected and ascertain certain correlations between specific data points collected in the last year from a fellow classmate's vehicle. These data points were processed, cleaned and imported into Google Colab where a machine learning algorithm was applied. Results from this work showed that the models we trained found a weak correlation between the speed and weather metrics.

CHAPTER 1: INTRODUCTION

Automobiles have become a pivotal part of society; the creation of the automobile changed the way transportation is conceptualized. Since their creation, automobiles have seen substantial growth in their onboard systems and overall computing capabilities, with features such as Global Positioning Services or GPS, media playback systems, and many others becoming mainstream and expected with the purchase of a new vehicle. These services use and create tons of digital data, which is then stored on the vehicle. This data has been used to create a whole new sector of forensics called vehicle forensics, which seeks to use the data collected by automobiles to provide new information services to consumers. The data can be used to do many different things from proving/disproving an alibi, to tracking wanted suspects' movements through GPS. There are programs developed for these specific purposes that law enforcement and other companies use.

In this paper, we develop an expandable dataset consisting of speed metrics and geo point markers mapped over a 15-day period as well as weather metrics collected from the times and places of the vehicle data. This data consists of historic vehicle data collected from August 15, 2023, to August 30, 2023, by a UNCW student who drives mainly to and from class as well as other daily chores/trips. This provides a consistent route over a daily period that would allow us to access it without too much variance. This data was collected using the Autopi CM4 Telemetric Fleet Management device which pulls data from the vehicles OBD-II port assuring accuracy and precision. This data is then sent to the Autopi hosted cloud, where it is catalogued and stored ready for project use. The weather data portion was collected using the Weatherbit historic weather API through the timestamp as well as geographic location of the vehicle data.

CHAPTER 2: Background Research

2.1 Vehicle Forensics

Vehicle forensics is a subcategory of forensics focused on automobiles and their systems. The field is thought to first have been applied with the use of skid marks to recreate accidents. Skid mark evaluation allowed investigators to recreate vehicle movements using simple comparisons between skid marks and known patterns. With this simple beginning, vehicle forensics would soon make huge advancements. With the invention of the On-Board Computer, the computing and recordkeeping capacities of automobiles have grown exponentially over the past decades. With the great technical advances made within automobiles, the number of sensors and overall data stored within automobiles has increased exponentially to the point that there is so much data, that a person's movements and many other facets of their life can be directly tracked just from the data gathered from their vehicle.

With this new era of vehicle data, there grew a need for vehicle forensics to be able to accurately use vehicle data for forensics purposes. With this need, there came the creation of many hardware and software solutions for vehicle forensics use. One of the main technologies used for vehicle forensics as of current is the BERLA software (Porter, 2021) and hardware toolset. The BERLA software and hardware toolset is a set of adapters and harnesses that allow for the extraction of certain memory chips within the automobiles computer system that allows their software suite to translate the data into usable data points that are then able to be assessed and combed through on the user's own computer system. The removal of the automobile's memory chips is called chip-off

(Gilware) and is quite common within law enforcement vehicle forensics as it is the most reliable form for data extraction from a vehicle.

2.2 Vehicle Forensics Tools

The main tool used by law enforcement for this purpose is the BERLA iVe toolkit, which is a hardware kit for data extraction as well as forensic software to make use of the data and represent it as readable and useful data. Another tool that law enforcement agencies use for vehicle forensics would be Blackbag's Mobilyze software (Tri Tech Forensics), which is a software tool that can access and display data from a vehicles infotainment system. This information would be mostly data collected from connected phones in more modern vehicles such as contacts and message history. Another tool that is used is ElcomSoft's iOS Forensic Toolkit (Co.Ltd.), which is an iOS tool that does a very similar job to the Blackbag software. The device we finally settled on for this project comes from a company called Autopi, which is an IoT company that started as three friends working on a tech project, that spiraled into a Kickstarter Campaign and was met with great success.

2.3 Machine Learning

Machine Learning is a subset of artificial intelligence that uses algorithms paired with datasets to gradually improve and assess patterns in data to improve predictions and overall accuracy. Within this project we used a handful of different machine learning models to achieve the project goal. These models are Linear Regression, Elastic Net, Ridge, Lasso, ExtraTreeRegressor, GradientBoostingRegressor, XGBRegressor, and LGBMRegressor. Each of these models have their own strengths and

weaknesses that they bring to the project, some proving more useful than others. Here we list a short description of each model and some of the advantages and disadvantages they each possess.

2.3.1 Linear Regression

Linear regression is one of the most basic forms of regression. In linear regression you have two variables, a dependent variable, and a predictor variable. These variables must be related linearly, which means they require a positive or negative relationship. Linear regression then finds a line of best fit for the data, mapping it and extrapolating other data points based off the slope, intercept, and error data of the line.

2.3.2 Elastic Net

Elastic Net is a regularized regression method that combines the features of both the Ridge and Lasso models. Which means it features the L1 penalty from Lasso as well as the L2 penalty from Ridge. It is a "best of both worlds" scenario in that regard.

2.3.3 Lasso

Lasso, or least absolute shrinkage and selection operator is a regularization technique that uses an L1 regularization technique. It is very useful when there is a high level of multicollinearity within the model. Regularization techniques are a powerful tool to avoid overfitting within data. Overfitting is an issue that arises when a model fits too well to training data and makes it difficult for the model to generalize to other data. In essence, it is when the model has been too closely fitted to training data in a way that new data does not generate properly and in essence makes the model useless for exploring new data. Regression that uses an L1 regularization technique means that it uses an L1

penalty. An L1 penalty means that a penalty is added to the model of the absolute value of the magnitude of the coefficient.

2.3.4 Ridge

Ridge is a regularization model like lasso, but instead of an L1 technique/penalty, it adds an L2 penalty which adds the square of the magnitude of coefficients.

2.3.5 ExtraTreeRegressor

ExtraTreeRegressor is an extremely randomized tree regressor from the scikit-learn machine learning library.

2.3.6 GradientBoostingRegressor

This is an "additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function" The loss function is a measure of the difference between an estimated value and the true value. Boosting refers to the type of machine learning model in which the repeated addition and fitting of decision tree models to data is used to correct prediction errors. Gradient Boosting is a technique of fitting a model using an arbitrary loss function and a gradient descent optimization algorithm. This causes the loss gradient to be reduced as a model is fit.

2.3.7 XGBRegressor

XGBRegressor or Extreme Gradient Boosting is a machine learning algorithm that uses the same gradient boosting the same as GradientBoostingRegressor, but it also uses a regularization technique to have control over how complex models are allowed to be.

2.3.8 LGBMRegressor

LGBMRegressor, or Light Gradient Boosting Machine is a gradient boosting framework that uses tree-based learning algorithms. It has fast-training speed, as well as better accuracy, and lower memory usage.

Though all these models have positives and negatives, they each play a role in developing a robust and functional machine learning algorithm. Once the models are trained, we will find which ones work best with our data set and focus on those ones.

2.4 Related Literature

There has been very little literature written about vehicle forensics and the processes related to it. Most of the papers that have been written about vehicle forensics have had access to and used the BERLA iVe system to pull data from the vehicles to do their research with. They, however, do not detail in any capacity the process of using the BERLA toolset. As far as the Autopi system is concerned, there have been no papers written using data pulled with Autopi.

CHAPTER 3: Proposed Problem

3.1 Initial Ideation

The goal of this paper is to retrieve vehicle data and then use this data paired with weather and speed metrics within a machine learning algorithm to determine if and to what degree a correlation exists between these metrics and if data can be accurately generated to reflect real world data. When doing research for this, it was discovered that vehicle data was not as easy to come by as one would first assume. Hours were spent scouring resources trying to find datasets, or any vehicle data that was available for

research purposes. All this research led to one place, BERLA, which is a company on the forefront of vehicle forensics. BERLA is used by law enforcement and anyone else in the field. They have developed a tool kit that allows the user to gather an incredible amount of vehicle data, from engine metrics to synced phones contact data. This large amount of data allows investigators to build robust portfolios for suspects based solely off their vehicle. Allowing investigators an unmatched look into a suspect's movements and even social life. Many people do not realize the caliber of data that is located on one's own vehicle.

3.2 Vehicle Data Unavailability

With all of this in mind, the next step was to reach out to BERLA and attempt to get access to their toolset. Upon reaching out to them, the response was given that they were not prepared for any work in the education sector. This in turn required research to be done into the area of gathering data from vehicles. With this new information, we shifted approaches and pursued Autopi, which is a fleet management device and cloud service built off the raspberry pi and intended for the use of vehicle management and data logging. We used the Autopi on a fellow student's vehicle for around a one-year length of time in which vehicle data was gathered anytime the vehicle was powered on. This data is available within Autopi cloud, which is their web hosted front facing platform. The data shown is quite limited and is displayed as "trips" or in other words it is shown in time slices that go from vehicle power on to vehicle power off. These "trips" do not allow a very robust view of the data that is collected, as it is hard to compare data collected over long stretches of time.

3.3 Where to find data?

With this knowledge we hooked an Autopi device to a fellow student's car which he then used as his daily driven vehicle for over a year, capturing all driving activity he did during that time. The data was then logged to the Autopi cloud where we were able to access it and view his trip data. This trip data was just not the preferred format for the data we would need to use within my machine learning algorithm, so we would need to find another access point for this data, which is where the Autopi api comes in. The Autopi cloud API is a REST API that accesses the Autopi cloud data using requests. With this API, we were able to make custom GET requests and receive whatever data I would need to work within my machine learning algorithm.

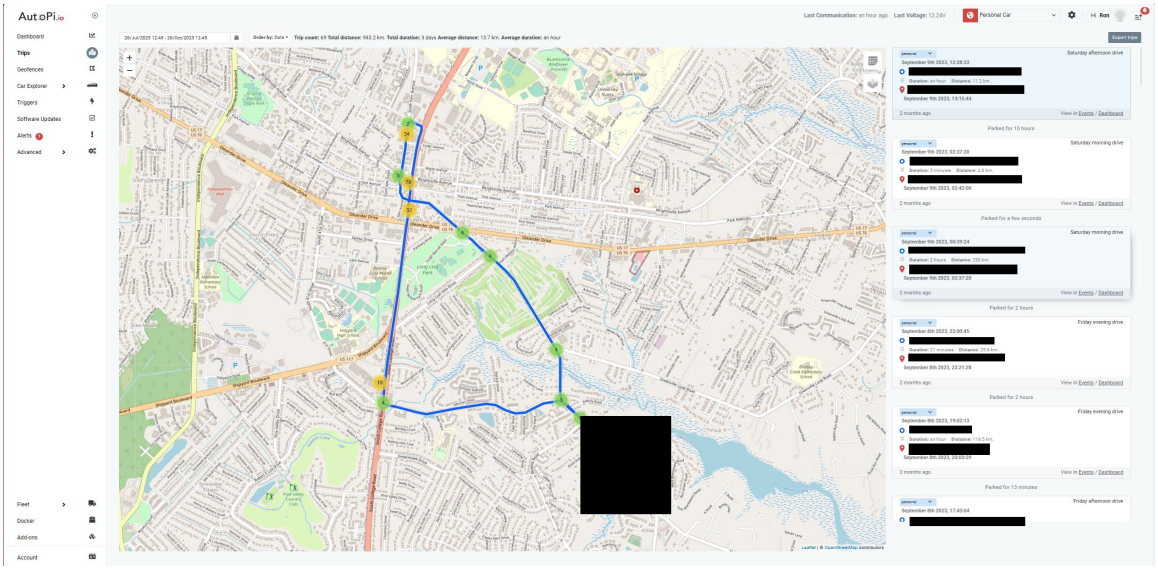


Figure 1 (Example of trip data from the autopi cloud with locations redacted to protect students' privacy.)

With the Autopi REST API, there are a few retrievable fields to choose from, we combed through them and removed many of the nonapplicable fields and created Table 1 to show all of the fields available with the data type and a short description of what they include.

Data Name	Data type	Description
obd.ambiant_air_temp.value	int	Ambient Air temperature
obd.bat.level	int	Charge level of car battery
obd.bat.state	string	State of car battery
obd.bat.voltage	float	voltage of car battery
obd.coolant_temp.value	int	vehicle coolant temp
obd.fuel_level.value	float	vehicle fuel level
obd.rpm.value	float	vehicle engine rpms
obd.speed.value	int	vehicle speed (kmph)
obd.temp_ext.value	float	exterior temperature
track.pos.alt	float	altitude
track.pos.cog	float	course over ground
track.pos.nsat	int	number of available satellites
track.pos.sog"	float	speed over ground
track.pos.loc	lat/long	latitude and longitude position

Table 1 field names, data type and description

These fields return the data with hard to decipher value names that will need to be translated into usable data points. The data will also need to be translated into a Pandas Dataframe format for use within the machine learning algorithm within Google Colab. I wrote a python program to take care of all these issues from making the request to the API to translating the returned data into a Dataframe with properly labeled headers. Once

the proper data is requested and converted to a Dataframe, this file is then uploaded into Google Colab, where the data was run through a simple exploratory analysis process to gain a clearer and more concise understanding of the data.

CHAPTER 4 – METHODOLOGY

4.1 Data Collection

The data needed to complete this project consists of a time metric for each record, a latitude for each record, a longitude for each record, a speed metric for each record, and then a medley of weather metrics ranging from temperature to precipitation. All these metrics were coordinated off the time metric, latitude, and longitude. The data was also saved as a Pandas Dataframe.

The vehicle data proved to be the largest time sink of the project, with us unable to locate a reliable source for vehicle data for many months and ultimately requiring us to record and extract our own data over the course of a year. This was achieved using the Autopi CM4 telemetric fleet management device, which is an OBD-II capable device built on the Raspberry pi platform and data connected using a SIM card. The device is capable of recording and transmitting OBD-II signals within the vehicles on board diagnostic system, which is then uploaded to the cloud that Autopi hosts and gives users access to. The data is then able to be accessed through Autpis REST API. This enables us to custom make requests and get data for whatever metrics or time frame we need, solving our vehicle data issue. A list of some of the data points that were estimated to be more valuable to this project can be found in Table 1 in the “Where to find data” section.

I created 2 GET requests within Postman one to retrieve the speed metric and one to get the Location data. When moving this to Python, I used the requests library within python to make the necessary requests. These Postman requests can be seen here: Speed Request: (Appendix C) Location Request: (Appendix D). All the data received from the different requests would need to be converted into Panda Dataframes, which would mean the requests would need to be received or translated into json() so that Pandas built in converter can take the values and build functional Dataframes from them.

The next task to tackle was acquiring weather metrics for the given timeframe using the location geo points. This was achieved using the Weatherbit API, which has a historic weather request, that takes latitude, longitude, and a date range and gives an hourly breakdown of weather metrics between the date range. I would have liked to get a more precise weather metric to work with, but that just was not feasible within the current project.

4.2 Data Sanitization

Now that we have all the data collected from separate API requests, we need to begin the process of combining and cleaning the data. This process requires the Dataframes the data is stored in to be merged. Unfortunately, the time formats are not the same between different API requests, so I had to write a program to translate them all to congruent date formats for them to be merged properly. The vehicle data from Autopi shared the same time format, which allowed easy merging of the vehicle data, though the weather metrics were based solely on an hourly format, which would require the times to be rounded to the nearest hour, I achieved this by creating a new field for “hourdate”, which is the date of the record rounded to the nearest floor hour. Due to the way Autopi

records their data, I ended up not having matching location data for some of my speed data, so I dropped all those records from the dataframe as they would not provide any use to us within the model training. There is a workflow showing the process the data went through from the raw data to the final dataset shown in Figure 2.

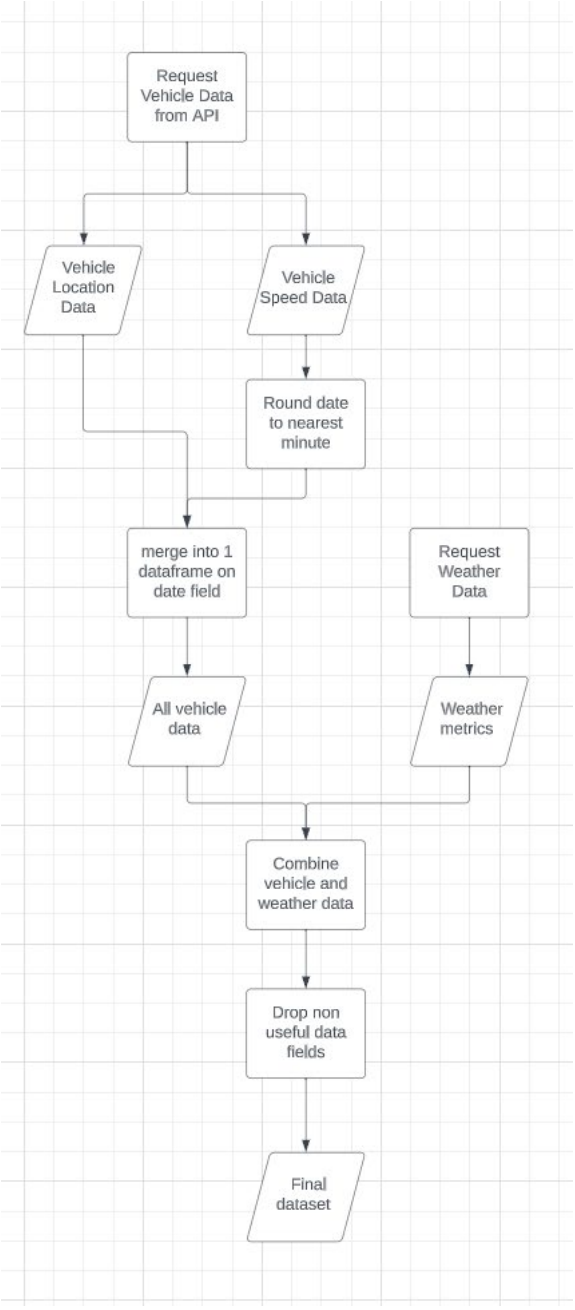


Figure 2 (Workflow describing the data cleaning process)

4.3 Data Description

With our data now cleaned and sorted, we are not able to begin doing some data analysis. One of the main functions of data analysis is to give perspective on what our data is and what it means in reference to our models and their values such as R^2, RMSE, and MAE. Below in Figure 3 we have the describe() method run on our data. This can be a good representation and breakdown of the different fields in our dataset and can also be a good visualization of similarities between fields.

The count field in the table displays the total number of records of that type within the dataset, as you can see from our table, we have 1623 of every field due to us dropping all NaN values in our cleaning step. The mean metric is the average value for each field, this shows us that the average relative speed variable was .611, which tells us in our data the speed was on average 61% of the speed average. The std metric displays the standard deviation of each field, which can be a good display of how much variety exists in our data. The min metric is the minimum value for each field, this can also be used with the max to show us the range of our data. The different percent metrics represent that percentile, essentially shows at what value for the field that percent of values falls under.

	rsv	precip	pres	rh	solar_rad	temp	uv	vis	wind_gust_spd	wind_spd
count	1623.000000	1623.000000	1623.000000	1623.000000	1623.000000	1623.000000	1623.000000	1623.000000	1623.000000	1623.000000
mean	0.611609	0.093277	1013.391251	72.899569	197.828096	28.073136	1.039248	15.622921	6.108872	5.305853
std	0.319238	0.340261	3.222967	10.754308	249.936896	1.010100	1.508126	1.456570	1.934098	1.874601
min	0.001782	0.000000	1009.000000	55.000000	0.000000	26.300000	0.000000	10.000000	1.500000	1.500000
25%	0.376045	0.000000	1010.000000	68.000000	0.000000	27.700000	0.000000	16.000000	5.100000	4.100000
50%	0.665558	0.000000	1014.000000	73.000000	92.000000	28.000000	0.600000	16.000000	6.000000	5.100000
75%	0.865777	0.000000	1015.000000	80.000000	498.000000	28.900000	1.800000	16.000000	8.000000	6.200000
max	1.159433	2.250000	1020.000000	95.000000	868.000000	29.800000	7.100000	16.000000	9.300000	9.300000

Figure 3 (The Describe() method run on our dataset)

We also have access to methods such as corr(), which shows us a table of correlation between every field in our dataset, this can be very helpful to determine which field are more valuable to us as well as debugging our model training. This table for our data can be seen in table 2 below. In this table, we can see the fields most correlated to our data are the humidity or RH and the dew point or DEWPT, these values have the most direct correlation as they are closer to one, which is a direct correlation.

index	rsv	clouds	dewpt	prec
rsv	1.0	-0.05762743708284224	0.14606644536753355	0.0752364
clouds	-0.05762743708284224	1.0	0.14370069082814188	0.1892616
dewpt	0.14606644536753355	0.14370069082814188	1.0	-0.0419
precip	0.07523640837105808	0.18926162903474125	-0.0419250494206	
pres	0.03371404323619532	-0.25837503455105626	-0.09552433114290039	-0.3737336
rh	0.1794164057629809	0.24083053609191443	0.9175747182891659	0.0030823
solar_rad	-0.11182415804436815	-0.22826622036421054	-0.49635664272157415	-0.0368475
temp	-0.03139513975463387	-0.218605621838488	0.3861587753869729	-0.1467105
uv	-0.049709261965578726	-0.20653888234161394	-0.34984057084025477	0.010672904
vis	-0.0325131090363604	0.06604696871823891	0.2588838332133453	-0.1392009
wind_gust_spd	-0.05036078264852145	-0.05797218645561824	0.221855235659554	0.172650
wind_spd	-0.03708186606517522	-0.16779053716400008	-0.019068660408402438	0.0689992

Table 2 (Displays the correlation data for our dataset)

4.3 Machine Learning Models Training

Now that the data was secured and sanitized as to be usable within machine learning models, we needed to determine a goof sampling of machine learning models to train using the data to get the best chance at being able to make useful estimates. The models we chose to use are Linear Regression, Elastic Net, Ridge, Lasso, ExtraTreeRegressor, GradientBoostingRegressor, XGBRegressor, and LGBMRegressor. These models were chosen as they give a wide range of different methods thus boosting our chances of having at least one that works decently.

Now that we have our list of models, they were all imported into Colab, which would allow us to use them. We then created a training set and a testing set of data using a random state of 75. The sets were created from the dataframe we previously created by dropping the metrics we were looking to estimate as well as any data that was a direct calculation or variant of the speed. We then created a pipeline in which each of the models would be trained using the training set. The code for this section is located in Appendix E.

4.4 Application of Models

The trained models were then applied to our dataset and predictions were created for each of the models creating a estimated data value for each timestamped data sample. This really allows us to see the power and accuracy of the models and how close they truly were to our actual values. This will be the piece that can be used with other sets of data and estimate the relative speed variable for those datasets. This is where the value is created in the project. The code for this area can be seen in Appendix F

CHAPTER 5: RESULTS

5.1 Expected Results

Going into this project, there were logical assumptions that could be made about the correlation between vehicle speed metrics and weather metrics. I had originally assumed that there would be a cut and dry correlation that would be visible just looking at the dataset. I had assumed that the weather data would have a lot more variety to it and be more distinct going day to day. When thinking of the weather, my mind immediately went to the time of year when you wake up in the morning and have to wear a sweater and by the time you get to work/school you are wishing you had shorts on. With all of this in mind, I assumed that the machine learning algorithm was going to be able to find a direct correlation between the vehicle and the weather data and from there be able to extrapolate and estimate data with ease.

5.2 Actual Results

After procuring the data and looking at it all connected and side by side, it became much clearer that the data was not going to have as much variance as I had initially assumed. I am not sure if it is a symptom of the time of the year the data was taken from (Summer, Eastern NA) or just a general misjudgment on my part, but hour to hour, the weather metrics did not change nearly as much as I had initially assumed. The limited time span of the data collection also likely played a part in the lack of variety of the data. From each of the trained machine learning models, we can ascertain 3 different metrics the R^2 value, the RMSE, and the MAE.

The R^2 value or the coefficient of determination is a measure of essentially how much correlation there is between an independent variable in the dataset and the one you

are looking to estimate. The R is the correlation coefficient, and it can be a very powerful tool in determining if correlation truly exists between a data set and metric. The RMSE or root mean square error is a measurement of the mean or average difference between the model's predicted value and the true value. In a more in-depth sense, it is the standard deviation of the distance between the regression line created by the model and the true data values. The final value, the MAE, is the mean absolute error, it is a measure of the average absolute value of the error within a model. If you are looking for a model that fits a dataset well, you want the model with the lower MAE. All the values for each of the trained models can be seen in the table below.

	R_Squared	RMSE	MAE
Linear	0.378306	0.242221	0.192918
ElasticNet	-0.014258	0.309383	0.265904
Ridge	0.381390	0.241619	0.194923
Lasso	-0.014258	0.309383	0.265904
Extra Tree	0.120590	0.288083	0.234633
Gradient Boosting	0.225058	0.270431	0.225305
XGradientBoosting	0.379141	0.242058	0.192452
LGBMRegressor	0.199059	0.274930	0.222809

Table 3 (displaying each model with their applicable R^2, RMSE, and MAE.)

With our models now trained, we were then able to make predictions for our Relative Speed Variable (RSV), which is a metric of the speed of the vehicle in mile per hour divided by the estimated speed limits of the roads that were driven on during the sample period. We could run our predictions through our previously created pipe and generate RSV values for each of our models and construct a full dataset of estimated relative speeds. A summation of that data can be seen in figure 4 and figure 5.

	time	rsv	Linear	ElasticNet	Ridge	Lasso	Extra Tree	Gradient Boosting	XGradientBooting	LGBMRegressor
0	2023-08-15:23	0.952769	0.862699	0.602433	0.834892	0.602433	0.670885	0.691027	0.786458	0.716986
1	2023-08-15:23	0.952769	0.952769	0.602433	0.907939	0.602433	0.670885	0.691027	0.786458	0.716986
2	2023-08-15:23	0.952769	0.840479	0.602433	0.818277	0.602433	0.670885	0.691027	0.786458	0.716986
3	2023-08-15:23	0.952769	0.952769	0.602433	0.907939	0.602433	0.670885	0.691027	0.786458	0.716986
4	2023-08-15:23	0.952769	0.952769	0.602433	0.886911	0.602433	0.670885	0.691027	0.786458	0.716986
...	...	...	...	...	...	...	...	...	...	...
1618	2023-08-27:20	0.004249	0.004249	0.602433	0.090166	0.602433	0.670885	0.398877	0.131064	0.714884
1619	2023-08-27:20	0.004249	0.004249	0.602433	0.090166	0.602433	0.670885	0.398877	0.131064	0.714884
1620	2023-08-27:20	0.004249	0.004249	0.602433	0.125081	0.602433	0.670885	0.398877	0.131064	0.714884
1621	2023-08-27:20	0.004249	0.049469	0.602433	0.183295	0.602433	0.670885	0.398877	0.131064	0.714884
1622	2023-08-27:20	0.004249	0.004249	0.602433	0.074259	0.602433	0.670885	0.398877	0.131064	0.714884

1623 rows × 10 columns

Figure 4 (Summation of trained models' estimations)

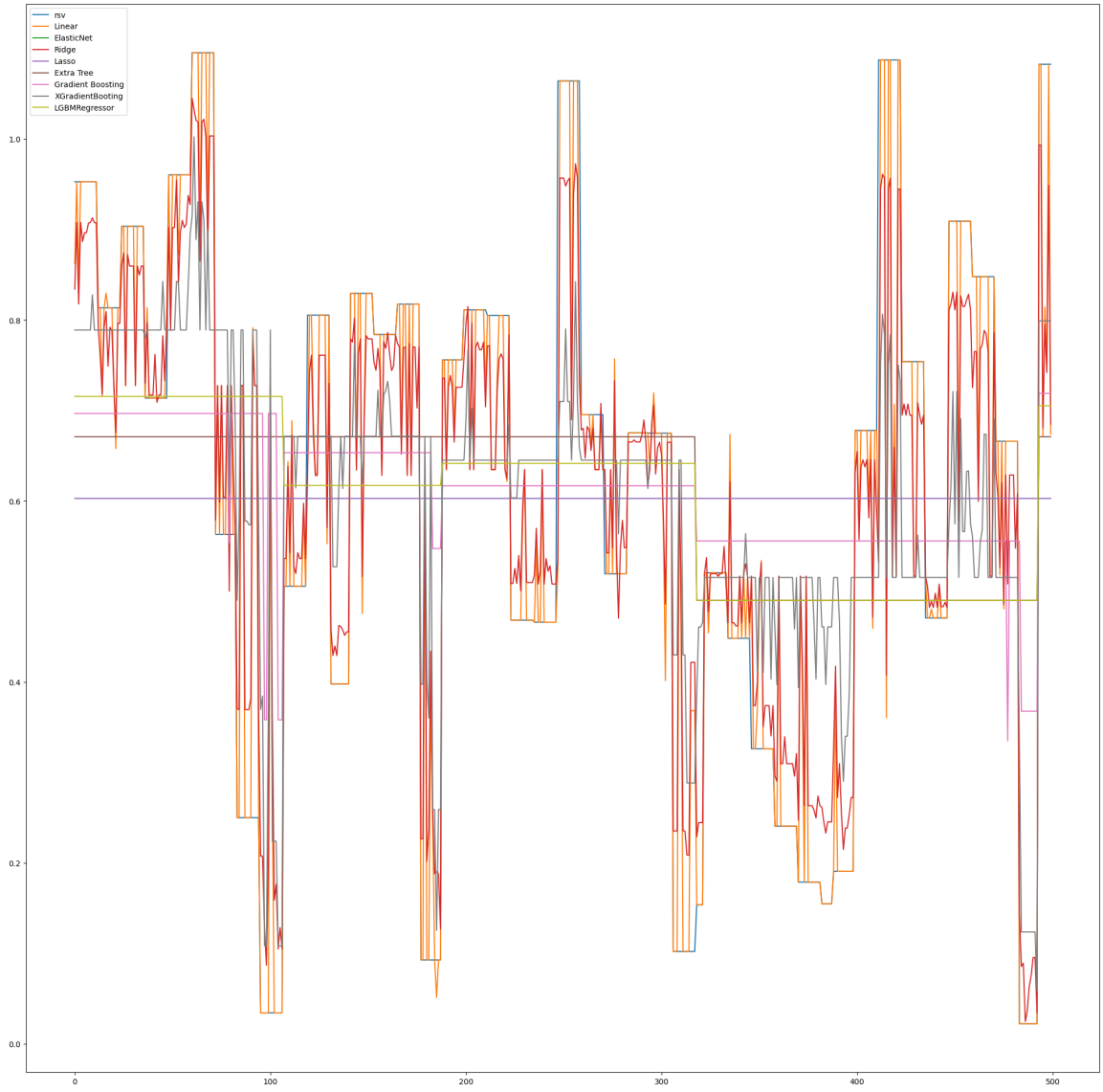


Figure 5 (Line plot of the first 500 records and their model estimations)

5.3 Assessment of Results

Having seen the results of the machine learning models, it became clear that our original assumptions about the dataset as well as the models proved to be quite on the mark. We had a pretty off the mark set of R^2, RMSE, and MAE across the board, with only Linear Regression, Ridge, and XgradientBoosting breaking a .3 R^2 value. This shows us that the dataset is not a good training set for our dataset, which I believe stems

from the lack of variation in weather conditions as well as pathing. The data was too similar, and thus the models had a hard time making accurate estimates and correct training. When viewing the data, it becomes quite apparent that Linear regression was the most accurate when it came to estimating the data,

CHAPTER 6: CONCLUSION

6.1 Issues

During the process of working on this project, we were met with many issues, stemming from problems finding a good solution for vehicle data acquisition to datasets not being a good fit for machine learning models. The first issue we faced was the difficulty in getting access to vehicle data. Our initial idea for gathering the data was to reach out to BERLA to try and get access to their iVe kit, the premier vehicle data acquisition tool. This tool set is a tool that even law enforcement use to gather data from vehicles for use in criminal cases. When we reached out, we were given a firm no in response which required us to seek other avenues and eventually settle on Autopi. The next issue we faced was actual being able to retrieve the data from the Autopi, we ran down the wrong path as far as trying to access the data from the vehicle and instead of using the API initially we were trying to query it from the device itself which proved to be way more difficult and unreliable than it was worth. Another issue with the Autopi system was the “clunkiness” of the API calls, the requests could only pull so much data with each request, and this hurt our ability to create a truly expansive and accurate

dataset. One of the bigger issues we ran into was the lack of access to paid APIs for our data needs. This required us to make estimates and overall reduce some of the accuracy of our dataset with the inability to get the speed limit for each section of road our vehicle traveled. All in all, the issues with the project were only either time wasters or wrong avenues other than the issue with not being able to get the speed limit data for road sections. So, in the long run they did not prove detrimental to the project.

6.2 Generated Value

This dataset and machine learning algorithm will prove very helpful as a steppingstone into the area of research. Many of the issues and suggestions made in this paper if properly applied to the research will create a robust and very valuable algorithm that can be used to provide suggested speeds for conditions on any driving scenario with a good bit of accuracy. These metrics can be used by either drives to better route trips, or even insurance companies/law enforcement to determine if a driver was going the appropriate speed for the conditions at the time of an accident. Most of the value for this project will come from the ability to estimate the relative speed variable that when applied to the speed limit of the road, will be a powerful tool to help drivers stay safe and flow more smoothly with traffic.

6.3 Reflection

Reflecting back on the project, there are a lot of things I would change in the beginning by pursuing Autopi from the start. This would have given us a lot more time to work with the device and built a much more varied and robust dataset that would work much better within the Machine learning models and be much more accurate with their

estimations. Another idea would be to take a stretch of road and have multiple people drive it in different weather conditions and use that as a more sanitized data sample. There are a lot of things that did not go right regarding API access as well, with some of the original ideas. We originally wanted to map each location to a road and then find the speed limit of that road, but unfortunately this was just not doable, as the only API that could achieve something like this was paid and required two location points which would not work with our dataset unfortunately. Other than these setbacks, the project worked pretty much as I had expected, and we found some results that kind of fit our initial assumptions. In summation, the project was not a complete failure, but did not have the pronounced and obvious conclusion I had originally hoped it would.

6.4 Future Works

Future works in this area will be expanding upon the groundwork put in place and broadening the dataset to include different times of the year instead of a single monotonous stretch of time. It would also help if for future works, they could incorporate multiple drivers to give even more data and not just mold the algorithm to one driver and instead hopefully make it more an all-encompassing fit.

REFERENCES

Nealon, L. (n.d.). *How many vehicles are there in the US?* Bankrate. Retrieved February 27, 2023, from https://www.bankrate.com/insurance/car/car-ownership-statistics/

Tire marks & their role in accident-reconstruction investigations. PInow.com - Find Local Investigators. (n.d.). Retrieved February 28, 2023, from https://www.pinow.com/articles/2399/tire-marks-their-role-in-accident-reconstruction-investigations

Porter, J. (2021, May 5). *Go read this report on how law enforcement can extract sensitive data from your car*. The Verge. https://www.theverge.com/2021/5/5/22420674/go-read-this-intercept-report-vehicle-forensics-infotainment-personal-sensitive-data-law-enforcement

Chip-off digital forensics services: Nand Recovery Services. Gillware. (n.d.). https://www.gillware.com/phone-data-recovery-services/chip-off-forensics-services/#:~:text=Chip-off%20forensics%20is%20an%20invasive%20forensic%20acquisition%20technique.,and%20properly%20realign%20their%20contents%20into%20coherent%20data.

Blackbag mobilyze software. Tri Tech Forensics. (n.d.). https://tritechforensics.com/blackbag-mobilyze-software/

Co.Ltd., E. (n.d.). *Elcomsoft IOS Forensic toolkit*. Elcomsoft Co.Ltd. https://www.elcomsoft.com/eift.html

Vadapalli, P. (n.d.). *6 types of regression models in Machine Learning You should know about*. upGrad blog. https://www.upgrad.com/blog/types-of-regression-models-in-machine-learning/

Hermawan, G., & Husni, E. (n.d.). Acquisition, modeling, and evaluating method of driving ... - iopscience. https://iopscience.iop.org/article/10.1088/1757-899X/879/1/012030

Chen, S.-H., Pan, J.-S., & Lu, K. (n.d.). Driving behavior analysis based on vehicle OBD information and AdaBoost. https://www.iaeng.org/publication/IMECS2015/IMECS2015_pp102-106.pdf

Rivera-Campoverde, N. D., Muñoz-Sanz, J. L., & Arenas-Ramirez, B. del V. (2021, September 23). Estimation of Pollutant Emissions in Real Driving Conditions Based on Data from OBD and Machine Learning. https://www.mdpi.com/1424-8220/21/19/6344

APPENDIX A

Example trip file download from Autopi

trips_data

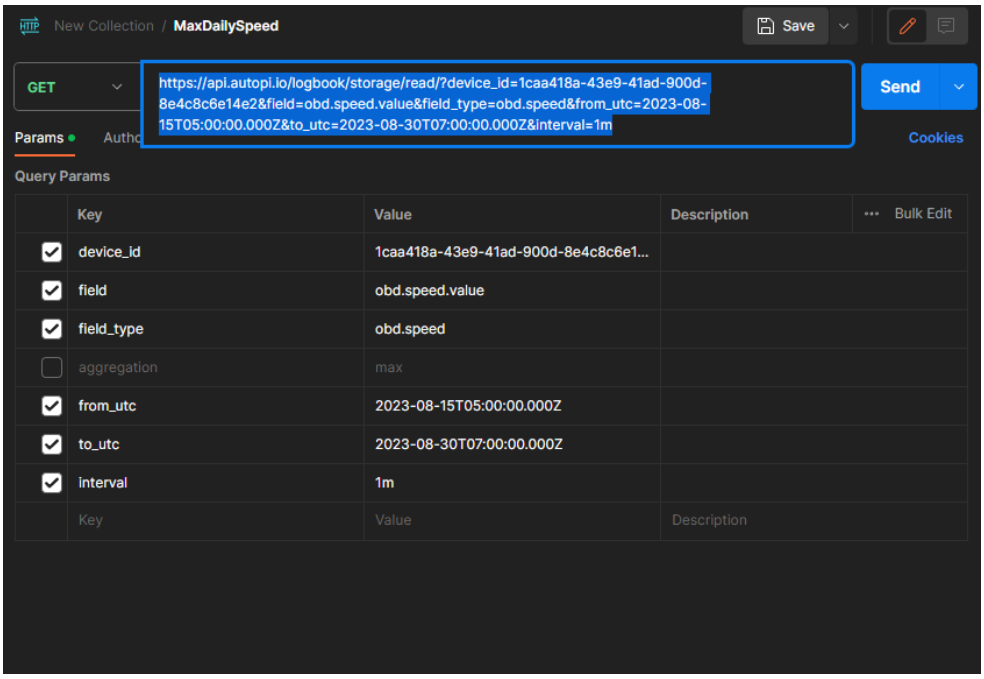
Start address	Distance travelled	Trip start time	Trip stop time	End address	Name	Mark trip personal/work
	0.00 km	2023-02-28T01:16:28.594Z	1970-01-01T00:00:00.000Z		evening drive	personal
239 Wilmington	16.72 km	2023-02-27T23:12:53.429Z	2023-02-28T01:16:28.593Z	Wilmington	evening drive	personal
242 Wilmington	10.11 km	2023-02-27T18:25:22.544Z	2023-02-27T23:12:53.428Z	239 Wilmington	afternoon drive	personal
Wilmington	7.93 km	2023-02-25T02:51:16.442Z	2023-02-27T18:25:22.543Z	242 Wilmington	evening drive	personal
242 Wilmington	6.97 km	2023-02-25T00:39:35.614Z	2023-02-25T02:51:16.441Z	Wilmington	evening drive	personal
4209 Wilmington	5.92 km	2023-02-24T02:09:43.717Z	2023-02-25T00:39:35.613Z	242 Wilmington	evening drive	personal
Wilmington	9.46 km	2023-02-23T23:40:08.014Z	2023-02-24T02:09:43.716Z	4209 Wilmington	evening drive	personal
5116 Wilmington	9.80 km	2023-02-23T20:35:46.101Z	2023-02-23T23:40:08.013Z	Wilmington	afternoon drive	personal
6505 Wilmington	6.34 km	2023-02-23T18:40:55.672Z	2023-02-23T20:35:46.100Z	5116 Wilmington	afternoon drive	personal
6450 Wilmington	6.36 km	2023-02-23T17:08:59.054Z	2023-02-23T18:40:55.671Z	6505 Wilmington	afternoon drive	personal
5202 Wilmington	7.02 km	2023-02-22T19:01:49.966Z	2023-02-23T17:08:59.053Z	6450 Wilmington	afternoon drive	personal
244 Wilmington	6.42 km	2023-02-22T17:14:21.974Z	2023-02-22T19:01:49.965Z	5202 Wilmington	afternoon drive	personal
5180 Wilmington	7.11 km	2023-02-21T20:35:28.384Z	2023-02-22T17:14:21.973Z	244 Wilmington	afternoon drive	personal
6450 Wilmington	6.50 km	2023-02-21T18:44:01.965Z	2023-02-21T20:35:28.383Z	5180 Wilmington		personal

APPENDIX B

Autopi CM4 Telemetric unit

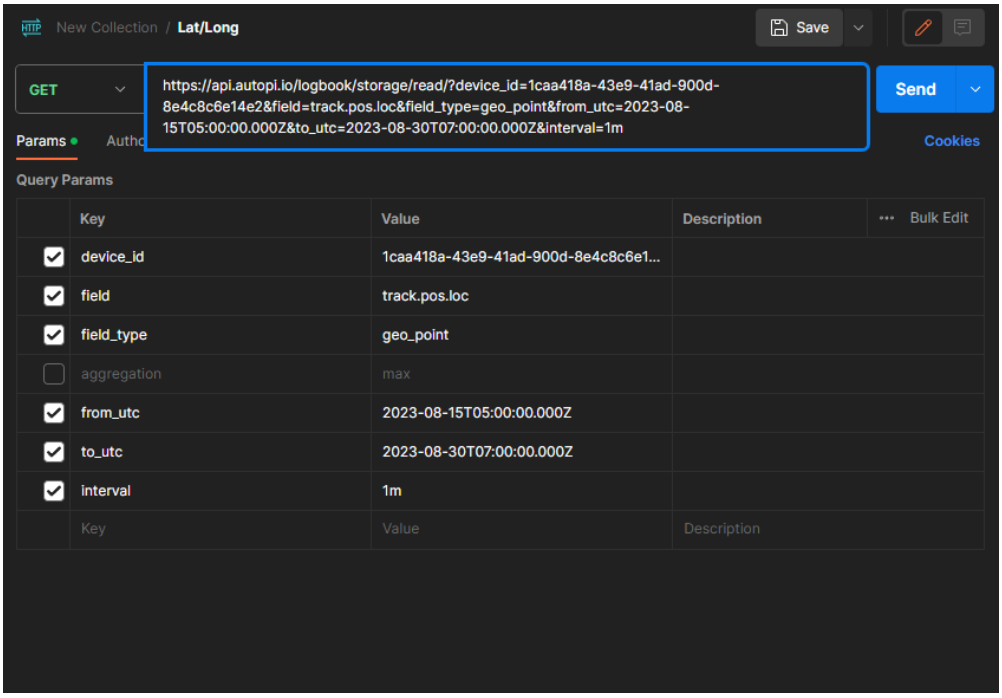
Autopi CM4 break down showing Internals

APPENDIX C



Postman GET request for the vehicle speed data between 08-15 and 08-30

APPENDIX D



Postman GET request for the vehicle location data between 08-15 and 08-30

Appendix E: Project code for generating the Dataset

```
with requests.Session() as s:
    p = s.post('https://api.autopi.io/auth/login', data=payload)
    response =
s.get('https://api.autopi.io/logbook/storage/read/?device_id=1caa418
a-43e9-41ad-900d-
8e4c8c6e14e2&field=obd.speed.value&field_type=obd.speed&from_utc=202
3-08-15T05:00:00.000Z&to_utc=2023-08-30T07:00:00.000Z&interval=1m',
headers={'Authorization': 'APIToken
7119ac871eb7875f6a5b353303f6686ddab80a20'})
    response2 =
s.get('https://api.autopi.io/logbook/storage/read/?device_id=1caa418
a-43e9-41ad-900d-
8e4c8c6e14e2&field=track.pos.loc&field_type=geo_point&from_utc=2023-
08-15T05:00:00.000Z&to_utc=2023-08-30T07:00:00.000Z&interval=1m',
        headers={'Authorization': 'APIToken
7119ac871eb7875f6a5b353303f6686ddab80a20'})

    outputlist = response.json()
    data1 = pd.DataFrame(outputlist)

    output2list = response2.json()
    data2 = pd.DataFrame(output2list)

    data2.dropna()
    location = pd.DataFrame(data2['location'])
    data2['lat'] = location['location']
    data2['lon'] = data2['location']
    data2['cog'] = data2['location']

    for i, row in data2.iterrows():
        data2['lat'][i] = data2['location'][i]['lat']
        data2['lon'][i] = data2['location'][i]['lon']
        data2['cog'][i] = data2['location'][i]['cog']

    data2 = data2.drop(['location'], axis=1)
    data1['ts'] = data1['ts'].str[:-7]
    data2['ts'] = data2['ts'].str[:-10]
    data1 = data1[data1.value != 0.0]
    output3 = pd.merge(data1,data2,
                       on='ts',
                       how='left'
                       )
```

```
    output3 = output3.drop(['max_ts'], axis=1)
    output3 = output3[~output3['value'].isnull()]
    output3 = output3[~output3['lat'].isnull()]
    output3['mph'] = output3['value'] * 0.62137119
    output3['rsv'] = output3['mph']/45
    output3['hourdate'] = output3['ts'].str.replace("T", ":").str[:-
4]


    with requests.Session() as s:
        weather =
s.get("https://api.weatherbit.io/v2.0/history/hourly?lat=34.18891166
666667&lon=-77.86393333333334&start_date=2023-08-15&end_date=2023-
08-28&tz=local&key=185325a6be074d9f9e6334329c086f7c")
        weatherlist = weather.json()['data']
        df = pd.DataFrame(weatherlist)
        output3 = pd.merge(output3, df,
                           left_on='hourdate', right_on='datetime',
                           how='left'
                           )
    output3 = output3.drop(columns=['ghi',
'azimuth','dhi','dni','elev_angle','slp','snow','solar_rad','ts_y','
wind_dir','wind_spd'], axis=1)
    output3.to_csv('output3.csv', sep='\t')
```

Appendix F: Project Code for model training

```
X = output3.drop(columns=['rsv'], axis=1)
y = output3['rsv']
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42)
L = LinearRegression()
E = ElasticNet()
R = Ridge()
Lass = Lasso(alpha=.1, max_iter=1)
ETR=ExtraTreeRegressor(max_depth=16, max_leaf_nodes=4)
GBR=GradientBoostingRegressor(loss='huber', learning_rate=.1,
n_estimators=50, max_leaf_nodes=3)
XGBC= XGBRegressor()
#cbst=CatBoostRegressor()
lgbm=LGBMRegressor()

algos = [L,E,R,Lass,ETR,GBR,XGBC,lgbm]
```

```
pipe = []
for algo in algos:
  pipe.append(make_pipeline(preprocessor, algo))

algo_names = ['Linear','ElasticNet','Ridge','Lasso','Extra
Tree','Gradient Boosting','XGradientBooting','LGBMRegressor']
r_squared = []
rmse = []
mae = []

result = pd.DataFrame(columns = ['R_Squared','RMSE','MAE'],
                      index = algo_names)

#result.index.name = 'Algorithms'

for pip in pipe:
    pip.fit(X_train,y_train)
    pip.predict(X_test)

    r_squared.append(r2_score(y_test,pip.predict(X_test)))
    rmse.append(mean_squared_error(y_test, pip.predict(X_test))**.5)
    mae.append(mean_absolute_error(y_test, pip.predict(X_test)))


result.R_Squared = r_squared
result.RMSE = rmse
result.MAE= mae
```

Appendix G: Application of models

```
predictions = pd.DataFrame(columns =
['time','rsv','Linear','ElasticNet','Ridge','Lasso','Extra
Tree','Gradient Boosting','XGradientBooting','LGBMRegressor'])

models = ['Linear','ElasticNet','Ridge','Lasso','Extra
Tree','Gradient Boosting','XGradientBooting','LGBMRegressor']

predictions.time = output3['hourdate']
predictions.rsv = output3['rsv']

for i in range(len(models)):
  predictions[models[i]] = pipe[i].predict(output3)
```

```
predictions
```