PREDICTION OF FERRY ARRIVAL TIMES USING MACHINE LEARNING

Mehmet Duman
University of Massachusetts, Dartmouth
mtduman@gmail.com

Faculty Advisor
Dr. David Koop

Abstract

In this project, we study machine learning methods to analyze publicly available data for Washington State Ferries' vessel trajectories. We studied both on-time and delayed trips, generated predicted arrival times, and compared our predictions with Washington State Ferries' own estimate time of arrival (ETA). The project used Python and the SciKit-Learn (sklearn) library which contains a variety of algorithms that can generate models and evaluate their predictive ability. The main contributions of the project are an analysis of the data retrieved from WSF website for the first quarter of 2017 and the predictions derived from different models created using linear regression, k-nearest neighbors, and random forest regression. We compared the results of the models to discover which method produces the best predictions in addition to comparing with the posted ETA prediction. Our study shows both that WSF's own ETA prediction can have significant errors, and that random forest regression gives the most accurate prediction of all the models tested.

Key words: Estimated Time of Arrival (ETA), Washington State Ferries (WSF), Vessel, Python SciKit-Learn library (Sklearn), Random Forest Regression, Linear Regression, K-Nearest Neighbors Regression.

TABLE OF CONTENTS

1. Introduction

The use of algorithms to calculate estimated arrival times has become widely popular because, in many aspects of our daily lives we have to be punctual. Mostly because we need to plan, schedule our times more than ever before. Accurate estimation of arrival times would economically, socially, and personally affect those involved.

In this study, we decided to use the Washington State Ferries (WSF)' data as our data source because, by serving eight counties in Washington and sailing international waters to Vancouver Island, British Colombia; WSF is the nation's largest ferry system. It is considered a marine highway, transporting thousands of commuters, students, commercial shippers and tourists across Puget Sound [21]. As of 2016 annually, approximately 10.5 million vehicles, and 24.2 million riders rely on ferry transportation for their daily routines [22]. It has very detailed and informative real-time data available to consume, and because of that it is suitable for our study.

As WSF grows over the years, the need for improved operations become imminent. The driving factors for these changes have been frequency, and convenience [2]. WSF did some analysis to find out what factors influence on-time performance of operations of the Ferries Division [4]. Among all factors accurate departure and arrival time information is one of the most important for the passengers and vehicle owners. As technological innovations increased, WSF did understand its customers' needs, and implemented its own departure time and estimated arrival time (ETA) and this is currently available on their site WSDOT [20].

In this project, we aim to analyze publicly available data from WSF [19], and try to find better prediction results than ETA for ferry arrival times by using and comparing three different SciKit-Learn regression analysis methods; Linear Regression, K-Nearest Neighbors Regression, and Random Forest.

This paper is structured in 7 sections. After the introduction, the remainder of the paper contains information about an overview of the existing literature (Section 2) and the related data (Section 3). Section 4 contains the methods we used, and in Section 5 and 6 the results we achieved, along with the limitations and obstacles we faced. Finally, in Section 7, we concluded our paper by presenting the major contributions.

2. Related Work

There are number of studies utilizing the machine learning techniques to the trajectory systems in many applications including ETA and /or trajectory calculations. Perera at al. [13] implemented machine learning technique (i.e. Kalman Filter) to predict the vessel trajectories. Valsamis et al. [17] tried to predict the location of the vessels in a time period by using many of the traditional machine learning algorithms and tested their model in real-time response setting with time-series data streams. In addition to these studies, there are two other works that used machine learning approaches with data from WSF to visualize the ferry pattern (Shanbhag et al. [14]) or to predict the ETA of vessel arriving at a port of Rotterdam (Parolas et al. [5]) using Support Vector Machine (SVM) and Neural Networks (NN).

Although a few studies used the data from WSF, none of them especially focused on ETA prediction. There are some reports published by WSF itself [18], however none of them clearly identify how they calculate the ETA estimate, and mostly provide information about the delays as a result of departure time.

From the literature review, we can see that there is a lot room for improvement in the prediction of ETA at ferry ports.

3. The Data

3.1 Data Source

The raw data used for this project has been collected from WSDOT web site (at a resolution of 1 minute) [19], and between January 1, 2017 to March 17, 2017. It is in JavaScript Object Notation-JSON format.

3.2 Data Collection & Data Cleaning

For the time period selected, 102K JSON files retrieved. There were only 3 files omitted which had empty or missing information. Each JSON file was individually processed and data was consolidated into a single comma-separated values (csv) file for further processing (Program 1). The total volume of this file is 2.5 GB.

In our dataset we had active 22 vessels and 20 ferry docks (Figure 1). Original dataset had approximately 2.44M records. Most of these records were generated while vessels were just waiting for their schedule or not in service. After filtering variables `inservice` (if the ferry in service or not) was true, `headtext` `(`shows ferries head in string format) was not stopped, `aterm_abbrev` (shows the ferries arrival dock code) was not null and `n_time_trip` (calculated trip duration) was less than a 3-minute trip; the problem records were removed and there were 685K records left on the file.

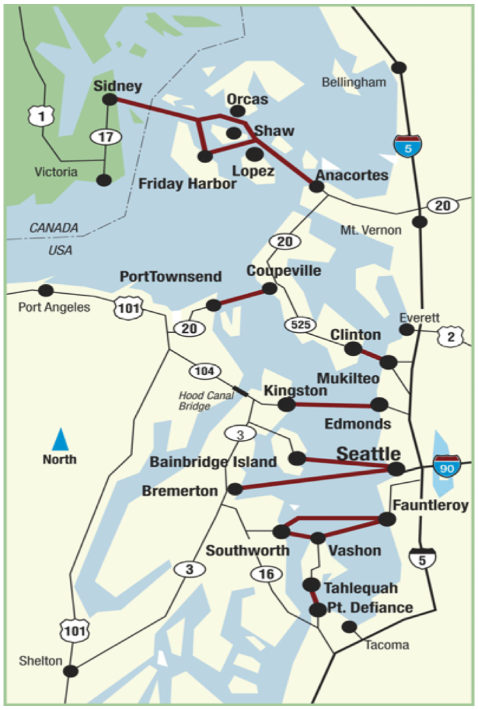


Figure 1

```
for infile_name in glob.glob('*.json'):
    with open(infile_name) as infile:
            try:
                    d                = json.load(infile)
                    df               = pd.DataFrame(d["vessellist"])
                    df['timestamp'] = d['timestamp']
                    if first:
                         df.to_csv(xOrigData, mode='w', header=True,
                         index=False)
                         first = False
                    else:
                         df.to_csv(xOrigData, mode='a', header=False,
                         index=False)
            except:
                    print("Problem on the file:" ,infile_name)
                    continue
```

Program 1

The dataset had many different date and time variables with different formats. Python's Pandas library has an applicable feature for date-time variables which you can convert any kind of date-time variable into a desirable format. In order to analyze and make predictions from the dataset, all those date-time variables had been converted to pandas date-time format (Year-month-day hour:minute:second) and new features generated for specific needs (Program 2). The list of all existing and new generated variables is in Appendix 1.

```
df['timestamp'  ]= pd.to_datetime(df['timestamp'], format="%m/%d/%Y %I:%M:%S   %p" )
df['n_datetime' ]= pd.to_datetime(df['timestamp'].dt.year.astype(str) + '/' +
                        df['datetime'].astype(str), format="%Y/%m/%d   %H:%M")
df['n_leftdock' ]= pd.to_datetime(df['n_datetime'].dt.date.astype(str)+ ' ' +
                        df['leftdock']+df['leftdockAMPM'], format="%Y-%m-%d %I:%M%p")
df['n_time_past']= ((df['n_datetime']-df['n_leftdock']) /
                        np.timedelta64(1, 'm')).astype(int)
df               = df[ df.n_time_past != -1 ]
df.loc[df.n_leftdock > df.n_datetime,'n_leftdock'] = df.n_leftdock-np.timedelta64(1,'D')
df['n_time_past']= (df['n_datetime'] –
                        df['n_leftdock']).astype('timedelta64[m]').astype(int)+1
df['n_time_trip']= df.groupby(['vesselID','n_leftdock',
                        'lastdock_id'])['n_time_past'].transform(max)+1
df['n_arrive']   = df.groupby(['vesselID','n_leftdock','lastdock_id'])
                        ['n_datetime'].transform(max)+np.timedelta64(1,'m')
```

Program 2: Converting to date-time (Y-m-d H:M:S)

For this dataset, we do not have any information for arrival time. Trips' last records of `datetime` (date-time of records generated) variable helps us to define the arrival time. There were some trips which had last records ending way before the arrival dock `aterm`. For those trips, it meant that we were calculating the wrong arrival time. Those trips defined as a missing trip. To remove those missing trips, the distance between trip's last location and `aterm` arrival dock calculated (Program 3) by Haversine Formula (Formula 1) [15]. From this `DistTripLastLocTOAterm` information, if a trip had a distance longer than 1000 meters, it meant that this trip ended more than 1000 meter before arrival dock. There were 14K records found like that and removed, which 10K missing trip records (Appendix 2) belonged to only 6 routes.

There were 3.8K records that had wrong route information or the vessel just waited at the same location for a long time which were not regular trip records. Those records were also removed from the dataset (Appendix 3). After all data were cleaned and problem records were removed, we had 667K records belonging to approximately 29.5K trips left in our dataset.

$$Haversine\ Dist = 2\,r\,\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + cos(\varphi_1)\,cos(\varphi_2)\,\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \qquad (1)$$

```
df['DistAterm'] = 6371 * 1000 * 2 * np.arcsin(
    np.sqrt(
        np.sin((  np.radians( df['AT_lat'] ) -np.radians(df['lat'] ))/2)**2
        + np.cos( np.radians( df['lat']    ))
        * np.cos( np.radians( df['AT_lat'] ))
        * np.sin((np.radians( df['AT_lon'] ) -np.radians(df['lon']))/2)**2))
df['DistTripLastLocTOAterm'] = df.groupby(['vesselID','n_leftdock'])
            ['DistAterm'].transform('last')
```

Program 3: Distance calculation with Haversine Formula

4. Methodology

In this section we try to describe the methodology we followed for developing our prediction model. For all models, we predict `n_time_left` (`TimeLeft;` time until arrival). Python program and its libraries Pandas, Sklearn and Matplotlib played a major role in thoroughly comprehending the dataset. In order to develop our model, we followed these steps:

4.1 Generating Features

New features were generated to use in fitting data and get better prediction for `n_time_left`. Here are some features generated;

`SpeedAvg`	Route speed average
`n_SpeedTripAvg`	Trip speed average
`n_SpeedCumAvg`	Trip speed cumulative average
`RunStdDev`	Trip running standard deviation
`TimeStampInSec`	The time of the day by seconds when record generated
`DistLastdock`	From current location to last dock Haversine distance [15]
`DistAterm`	From current location to arrival dock aterm Haversine distance [15]
`n_dist_past`	Trip cumulative distance the ferry past

4.2 Model Development

After retrieving, cleaning, and converting the data to a proper format and generating new features; we started our model development. There was not any information for arrival time, so this is calculated by using the trip last record `n_datetime` (date-time information that record generated) variable and adding one minute to estimate real arrival time. Next, we selected the input variables to fit to the model;

`DistAterm`	From current location to arrival dock aterm Haversine distance [15]
`speed`	Ferries current speed
`n_time_past`	Trip time past in minutes since started
`RunStdDev`	Trip running standard deviation
`TimeStampInSec`	The time of the day by seconds when record generated

After this we used the following three different machine learning regressions to predict `n_left_time` (`TimeLeft;` time until arrival). Machine learning technique explores the study and construction of algorithms that can learn from and make predictions on data [3]. A regression builds a relationship between an independent and a dependent variable (`n_left_time`) by using the historical data. Regression uses this relationship to make a prediction of the feature values of the dependent variable in a new dataset. It is important to compare the performance of multiple different machine learning algorithms consistently. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data [1]. Meaning, we fit the same part of data set to all regressions in the same way. Table 1 shows the comparison between three regression methods used in this study. From the score comparison results; we decided to continue our analysis with Random Forest method. These results will be discussed in the Findings section. Following are the regression models we used for this project:

4.2.1 Linear Regression

It fits a linear model to the data set by adjusting a set of parameters in to make the sum of the squared residuals of the model as small as possible [6].

Linear models : $y = X\beta + \varepsilon$

X	: data
y	: target variable
β	: Coefficients
ε	: Observation noise

4.2.2 K-Nearest Neighbors Regression

K-Nearest Neighbors Regression are made by using entire training dataset directly. To predict a new feature (y), regression search the training dataset for the K most similar nearest neighbors. K-Nearest Neighbors Regression implements learning based on the K nearest neighbors of each query point, where K is an integer value specified by the user. For this regression, we used following parameters [7];

`n_neighbors` = 200 : Number of neighbors to use by default for
k neighbors queries (default = 5)

`weights` = 'distance' : Weight function used in prediction 'distance'; weight
points by the inverse of their distance. In this case,
closer neighbors of a query point will have a greater
influence than neighbors which are further away

4.2.3 Random Forest Regression

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. Every observation in the dataset is fed into every decision tree. The most common outcome for each observation is used as the final output. For this regression, we used following parameter [8];

`n_estimators=500` : the number of trees in the forest (default=10)

`oob_score =True` : whether to use out-of-the-bag samples to estimate
the R^2 on unseen data

`max_features=None` : the number of features to consider when looking
for the best split (default='auto')

`n_jobs=-1` : the number of jobs to run in parallel for both fit
and predict. If -1, then the number of jobs is set to the
number of cores (default=1)

Following program codes used for regression predictions and errors calculations (Program 4):

```
x_model = [(LinearRegression(),
        'LR_predict','LRscore','LRcorrcoef','LRmae','LRrmse','LRevscore','LRmape','LRsmape','LRptime',''),
(KNeighborsRegressor(n_neighbors=200, weights='distance'),
        'KN_predict','KNscore','KNcorrcoef','KNmae','KNrmse','KNevscore','KNmape','KNsmape','KNptime',''),
(RandomForestRegressor(n_estimators=500,oob_score=True, max_features=None, n_jobs=-1 ),
       'RF_predict','RFscore','RFcorrcoef','RFmae','RFrmse','RFevscore','RFmape','RFsmape','RFptime','RFfeatImp'),
('ETA'
       ,''        ,'ETscore','ETcorrcoef','ETmae','ETrmse','ETevscore','ETmape','ETsmape','ETptime',''   ) ]

X_train = trdf[ ['DistAterm','speed','n_time_past','RunStdDev', 'TimeStampInSec'] ]
X_test  = tsdf[ ['DistAterm','speed','n_time_past','RunStdDev', 'TimeStampInSec'] ]
y_train = trdf['n_time_left']
y_test  = tsdf['n_time_left']
for ind, model in enumerate(x_model):
    if (model[0] == 'ETA'):
        y_p                              = tsdf['etaTimeLeft']
        xdf_r.loc[xind, x_model[ind][9]] = 0.0
    else :
        t_start = time.time()
        model[0].fit(X_train, y_train)
        y_p                              = model[0].predict(X_test)
        tsdf[ x_model[ind][1] ]          = y_p
        xdf_r.loc[xind, x_model[ind][9]] = time.time() - t_start

    s       = round( r2_score(y_test, y_p)                          , 3)
    c       = round( np.corrcoef(y_test, y_p )[0,1]                 , 3)
    mae     = round( mean_absolute_error(y_test, y_p)               , 3)
    rmse    = round( np.sqrt( mean_squared_error(y_test, y_p) )     , 3)
```

Program 4: Regressions and error calculations

4.3 Error Algorithm Used for Evaluating ETA Prediction

These are the error algorithms used for evaluating ETA prediction:

Score : R^2 coefficient of determination, a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse) [9].

Corrcoef : Return Pearson product-moment correlation coefficients. The values of R are between -1 and 1, where 0 is no relation, 1 is very strong, linear relation and -1 is an inverse linear relation [16].

MAE : The Mean Absolute Error calculate, a risk metric corresponding to the expected value of the absolute error loss [10], simply look at the "average difference" between those twos values.

RMSE : Root Mean Squared Error computes, a risk metric corresponding to the expected value of the root mean squared error regression loss [11]. It represents the similarity between those two values. It is taking square root of mean squared.

5 Findings

In this section, the results from the application of the methodology described above are presented.

5.1 Regression Model Scores and Process Time

Table 1 shows the comparison between three regression methods used in this study. Among the three methods tested, Random Forest Regression model gives the highest scores most of the time. However, the Train/Predict Duration time for Random Forest is much higher than the other two. One of the reason for this is, the number of trees in the Random Forest is 500 (`n_estimator` default is 10) to get better accuracy. Since Random Forest prediction scores are much better than the Linear Regression and K-Neighbors, we are going to compare this results with ETA.

5.2 Random Forest `feature_importance_`

After deciding to use Random Forest, we decided to fit some of the new features into the model. Sklearn Random Forest has great option called `feature_importance_`. This shows us, the use of forests of trees to evaluate the importance of features on a regression task [12]. We tried `feature_importance_` tool to see if it is going to make any difference in prediction and score. Some features were significantly effective to improve the prediction result, but most of them not. In the regression model, using more features increased the score at least 0.001 for most of the variable (Table 2).

			Linear Regression		K-Neighbors Regression		Random Forest Regression		Random Forest Regression `Feature_Importances_`	
					n_neighbors=5, weights='uniform'		n_estimators=500 (The number of trees in the forest) max_features=None (The number of future to consider) oob_score=True (whether to use out-of-bag samples) n_jobs= -1 (The number of jobs to run in parallel)			
Route	**Train Data Size**	**Test Data Size**	**Score**	**Train Predict Duration**	**Score**	**Train Predict Duration**	**Score**	**Train Predict Duration**	**Score**	**Train Predict Duration**
ANA-FRH	3,298	12,210	**0.978**	0.00"	**0.983**	0.44"	**0.987**	2.47"	**0.984**	4.28"
ANA-LOP	4,279	15,389	**0.982**	0.00"	**0.977**	0.75"	**0.981**	3.65"	**0.972**	5.36"
ANA-ORI	1,761	5,221	**0.984**	0.00"	**0.969**	0.20"	**0.979**	1.52"	**0.979**	3.28"
ANA-SHI	1,747	3,577	**0.984**	0.00"	**0.961**	0.14"	**0.985**	1.52"	**0.981**	3.12"
BBI-P52	11,048	37,197	**0.981**	0.00"	**0.981**	1.47"	**0.987**	7.69"	**0.977**	11.38"
BRE-P52	12,209	41,780	**0.983**	0.00"	**0.990**	1.51"	**0.993**	8.60"	**0.982**	12.90"
CLI-MUK	7,216	24,762	**0.935**	0.00"	**0.941**	0.92"	**0.963**	4.68"	**0.959**	8.23"
EDM-KIN	8,107	27,591	**0.963**	0.00"	**0.969**	1.06"	**0.978**	5.41"	**0.972**	8.81"
FAU-SOU	1,284	4,763	**0.890**	0.00"	**0.883**	0.16"	**0.917**	1.24"	**0.911**	2.80"
FAU-VAI	7,192	24,476	**0.914**	0.00"	**0.909**	0.91"	**0.938**	4.78"	**0.938**	9.12"
FRH-ANA	3,648	12,441	**0.971**	0.00"	**0.962**	0.42"	**0.982**	2.81"	**0.981**	5.39"
FRH-LOP	1,748	4,810	**0.938**	0.00"	**0.937**	0.16"	**0.957**	1.49"	**0.957**	3.27"
FRH-ORI	1,999	8,551	**0.924**	0.00"	**0.917**	0.32"	**0.956**	1.73"	**0.952**	3.65"
FRH-SHI	404	1,612	**0.967**	0.00"	**0.904**	0.04"	**0.984**	0.90"	**0.980**	2.02"
KEY-POT	3,695	14,672	**0.977**	0.00"	**0.968**	0.50"	**0.981**	2.63"	**0.978**	4.54"
KIN-EDM	7,998	27,634	**0.960**	0.00"	**0.967**	1.12"	**0.974**	5.37"	**0.969**	8.56"
LOP-ANA	4,754	17,211	**0.973**	0.00"	**0.970**	0.61"	**0.980**	3.41"	**0.966**	6.07"
LOP-FRH	1,813	5,957	**0.954**	0.00"	**0.974**	0.20"	**0.984**	1.49"	**0.980**	3.14"
LOP-ORI	269	1,108	**0.960**	0.00"	**0.901**	0.03"	**0.973**	0.69"	**0.972**	1.99"
LOP-SHI	975	3,285	**0.937**	0.00"	**0.880**	0.11"	**0.967**	1.12"	**0.965**	2.58"
MUK-CLI	7,819	26,880	**0.924**	0.00"	**0.936**	1.02"	**0.956**	5.20"	**0.956**	10.04"
ORI-ANA	2,637	6,641	**0.978**	0.00"	**0.972**	0.23"	**0.985**	2.33"	**0.977**	3.75"
ORI-FRH	2,400	8,943	**0.957**	0.00"	**0.928**	0.31"	**0.977**	1.82"	**0.966**	3.65"
ORI-LOP	341	1,323	**0.874**	0.00"	**0.806**	0.03"	**0.880**	0.70"	**0.880**	1.99"
P52-BBI	10,936	37,668	**0.972**	0.00"	**0.976**	1.48"	**0.984**	7.81"	**0.967**	11.28"
P52-BRE	12,067	42,298	**0.973**	0.00"	**0.978**	1.58"	**0.987**	8.51"	**0.980**	13.66"
POT-KEY	3,721	14,049	**0.978**	0.00"	**0.971**	0.51"	**0.982**	2.61"	**0.976**	4.44"
PTD-TAH	3,528	12,127	**0.935**	0.00"	**0.887**	0.40"	**0.958**	2.38"	**0.958**	4.92"
SHI-ANA	496	1,840	**0.969**	0.00"	**0.961**	0.06"	**0.972**	0.91"	**0.967**	2.11"
SHI-LOP	1,275	4,432	**0.906**	0.00"	**0.888**	0.15"	**0.938**	1.27"	**0.937**	2.90"
SHI-ORI	957	2,798	**0.870**	0.00"	**0.670**	0.09"	**0.903**	1.01"	**0.902**	2.55"
SOU-FAU	2,126	7,564	**0.821**	0.00"	**0.893**	0.26"	**0.912**	1.72"	**0.912**	3.49"
SOU-VAI	2,930	9,798	**0.917**	0.00"	**0.862**	0.32"	**0.941**	2.06"	**0.941**	4.28"
TAH-PTD	3,577	11,991	**0.937**	0.00"	**0.904**	0.41"	**0.964**	2.40"	**0.964**	4.92"
VAI-FAU	6,631	22,638	**0.872**	0.00"	**0.877**	0.81"	**0.916**	4.81"	**0.915**	8.69"
VAI-SOU	3,329	10,914	**0.887**	0.00"	**0.868**	0.35"	**0.936**	2.39"	**0.936**	4.85"

Table 1

			Random Forest with Sklearn Feature Importances						Random Forest								
			Input Features: `Feature_Importance_`						Input Features: `DistAterm, TimeStampInSec` `Speed, TimePast, SpeedRunDev`								
			Predict : `LeftTime`						Predict :`LeftTime`					ETA			
Route	Train Data Size	Test Data Size	Score	Corr. Coeff.	Mean Absolute error	Root mean squared error	Train Predict Dur.	Feature_Importance_ SelectFromModel(clf, threshold=0.01)	Score	Corr. Coeff.	Mean absolute error	Root mean squared error	Train Predict Duration	Score	Correlation coefficient	Mean absolute error	Root mean squared error
ANA-FRH	3,298	12,210	**0.984**	0.99	1.5	2.39	4.28"	`DistAterm,n_time_past`	**0.987**	0.99	1.21	2.15	2.47"	**0.87**	0.95	1.83	6.71
ANA-LOP	4,279	15,389	**0.972**	0.99	1.31	1.92	5.36"	`DistAterm`	**0.981**	0.99	1.02	1.58	3.65"	**0.78**	0.91	1.40	5.35
ANA-ORI	1,761	5,221	**0.979**	0.99	1.28	2.15	3.28"	`DistAterm,n_time_past,RunStdDev`	**0.979**	0.99	1.15	2.14	1.52"	**0.61**	0.86	1.75	9.18
ANA-SHI	1,747	3,577	**0.981**	0.99	1.19	1.94	3.12"	`DistAterm`	**0.985**	0.99	0.97	1.74	1.52"	**0.96**	0.98	1.57	2.87
BBI-P52	11,048	37,197	**0.977**	0.99	0.94	1.35	11.38"	`DistAterm`	**0.987**	0.99	0.68	1.03	7.69"	**0.97**	0.99	1.06	1.63
BRE-P52	12,209	41,780	**0.982**	0.99	1.42	2.09	12.90"	`DistAterm`	**0.993**	1.00	0.88	1.31	8.60"	**0.98**	0.99	1.58	2.23
CLI-MUK	7,216	24,762	**0.959**	0.98	0.53	0.80	8.23"	`DistAterm,speed,RunStdDev`	**0.963**	0.98	0.50	0.76	4.68"	**(131.99)**	0.19	5.18	45.89
EDM-KIN	8,107	27,591	**0.972**	0.99	0.76	1.09	8.81"	`DistAterm,speed`	**0.978**	0.99	0.67	0.98	5.41"	**0.94**	0.97	0.97	1.59
FAU-SOU	1,284	4,763	**0.911**	0.96	1.02	1.83	2.80"	`DistAterm,speed,RunStdDev`	**0.917**	0.96	0.99	1.77	1.24"	**0.80**	0.91	1.87	2.73
FAU-VAI	7,192	24,476	**0.938**	0.97	0.60	1.07	9.12"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.938**	0.97	0.60	1.07	4.78"	**(49.90)**	0.13	1.79	30.58
FRH-ANA	3,648	12,441	**0.981**	0.99	1.51	2.68	5.39"	`DistAterm,speed,RunStdDev`	**0.982**	0.99	1.49	2.65	2.81"	**0.96**	0.98	2.20	3.93
FRH-LOP	1,748	4,810	**0.957**	0.98	1.24	1.95	3.27"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.957**	0.98	1.22	1.94	1.49"	**0.86**	0.93	2.51	3.51
FRH-ORI	1,999	8,551	**0.952**	0.98	1.35	2.38	3.65"	`DistAterm,TimeStampInSec,speed`	**0.956**	0.98	1.26	2.29	1.73"	**0.85**	0.94	2.19	4.26
FRH-SHI	404	1,612	**0.980**	0.99	1.25	1.75	2.02"	`DistAterm,n_time_past`	**0.984**	0.99	1.06	1.53	0.90"	**0.99**	0.99	0.89	1.46
KEY-POT	3,695	14,672	**0.978**	0.99	0.81	1.20	4.54"	`DistAterm,speed`	**0.981**	0.99	0.75	1.11	2.63"	**0.95**	0.98	1.16	1.84
KIN-EDM	7,998	27,634	**0.969**	0.99	0.78	1.16	8.56"	`DistAterm,speed`	**0.974**	0.99	0.69	1.05	5.37"	**0.57**	0.84	1.11	4.32
LOP-ANA	4,754	17,211	**0.966**	0.98	1.55	2.26	6.07"	`DistAterm`	**0.980**	0.99	1.15	1.76	3.41"	**(15.00)**	0.21	3.73	49.14
LOP-FRH	1,813	5,957	**0.980**	0.99	0.99	1.41	3.14"	`DistAterm,speed,RunStdDev`	**0.984**	0.99	0.90	1.28	1.49"	**0.96**	0.98	1.53	2.08
LOP-ORI	269	1,108	**0.972**	0.99	0.60	0.82	1.99"	`DistAterm,speed,n_time_past`	**0.973**	0.99	0.58	0.80	0.69"	**0.68**	0.89	1.30	2.78
LOP-SHI	975	3,285	**0.965**	0.98	0.57	0.83	2.58"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.967**	0.98	0.57	0.82	1.12"	**0.86**	0.94	1.04	1.68
MUK-CLI	7,819	26,880	**0.956**	0.98	0.60	0.91	10.04"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.956**	0.98	0.60	0.91	5.20"	**0.81**	0.92	1.19	1.91
ORI-ANA	2,637	6,641	**0.977**	0.99	1.48	2.37	3.75"	`DistAterm,RunStdDev`	**0.985**	0.99	1.17	1.92	2.33"	**0.96**	0.98	1.72	2.96
ORI-FRH	2,400	8,943	**0.966**	0.98	1.33	1.99	3.65"	`DistAterm,n_time_past`	**0.977**	0.99	1.02	1.62	1.82"	**0.89**	0.95	1.97	3.61
ORI-LOP	341	1,323	**0.880**	0.94	1.39	2.06	1.99"	`DistAterm,TimeStampInSec,speed,n_time_past,RunStdDev`	**0.880**	0.94	1.39	2.06	0.70"	**0.69**	0.87	2.16	3.29
P52-BBI	10,936	37,668	**0.967**	0.98	1.18	1.66	11.28"	`DistAterm`	**0.984**	0.99	0.79	1.17	7.81"	**(1.60)**	0.51	1.50	14.66
P52-BRE	12,067	42,298	**0.980**	0.99	1.44	2.22	13.66"	`DistAterm,speed`	**0.987**	0.99	1.15	1.78	8.51"	**0.97**	0.99	1.70	2.56
POT-KEY	3,721	14,049	**0.976**	0.99	0.82	1.21	4.44"	`DistAterm`	**0.982**	0.99	0.67	1.05	2.61"	**0.96**	0.98	1.00	1.58
PTD-TAH	3,528	12,127	**0.958**	0.98	0.50	0.73	4.92"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.958**	0.98	0.50	0.73	2.38"	**(7.48)**	0.34	1.33	10.33
SHI-ANA	496	1,840	**0.967**	0.98	1.53	2.64	2.11"	`DistAterm`	**0.972**	0.99	1.18	2.43	0.91"	**0.94**	0.97	2.03	3.43
SHI-LOP	1,275	4,432	**0.937**	0.97	0.82	1.23	2.90"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.938**	0.97	0.81	1.22	1.27"	**(2.59)**	0.46	2.04	9.29
SHI-ORI	957	2,798	**0.902**	0.95	0.53	0.81	2.55"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.903**	0.95	0.52	0.81	1.01"	**(348.79)**	0.07	5.40	48.27
SOU-FAU	2,126	7,564	**0.912**	0.96	1.17	1.93	3.49"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.912**	0.96	1.16	1.93	1.72"	**0.48**	0.78	2.54	4.67
SOU-VAI	2,930	9,798	**0.941**	0.97	0.54	0.83	4.28"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.941**	0.97	0.54	0.82	2.06"	**0.48**	0.82	1.27	2.44
TAH-PTD	3,577	11,991	**0.964**	0.98	0.46	0.67	4.92"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.964**	0.98	0.46	0.67	2.40"	**0.08**	0.74	1.27	3.40
VAI-FAU	6,631	22,638	**0.915**	0.96	0.75	1.31	8.69"	`DistAterm,TimeStampInSec,speed,RunStdDev`	**0.916**	0.96	0.75	1.31	4.81"	**0.68**	0.86	1.62	2.56
VAI-SOU	3,329	10,914	**0.936**	0.97	0.57	0.88	4.85"	`DistAterm,TimeStampInSec,speed,n_time_past,RunStdDev`	**0.936**	0.97	0.57	0.88	2.39"	**(7.82)**	0.33	1.44	10.32

Table 2

As a result of our experiment we saw that, fitting only the `feature_importance_` variable to the Random Forest regression model, most of time gives us the same score as using chosen variables. At the same time, `feature_importance_` option uses the entire training data to calculate the informative variable, but it also increases the Train/Predict Duration Time almost twofold for the model (Table 2). As we can see from below Figure 2, even the reverse trips in same route, the `feature_importance_` variable may come out differently.

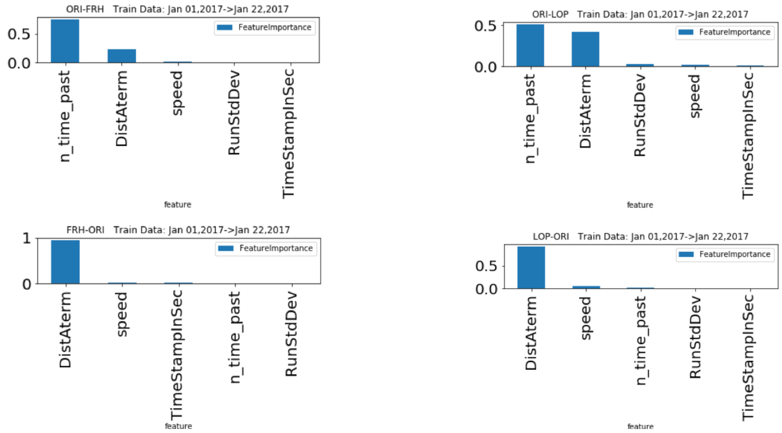
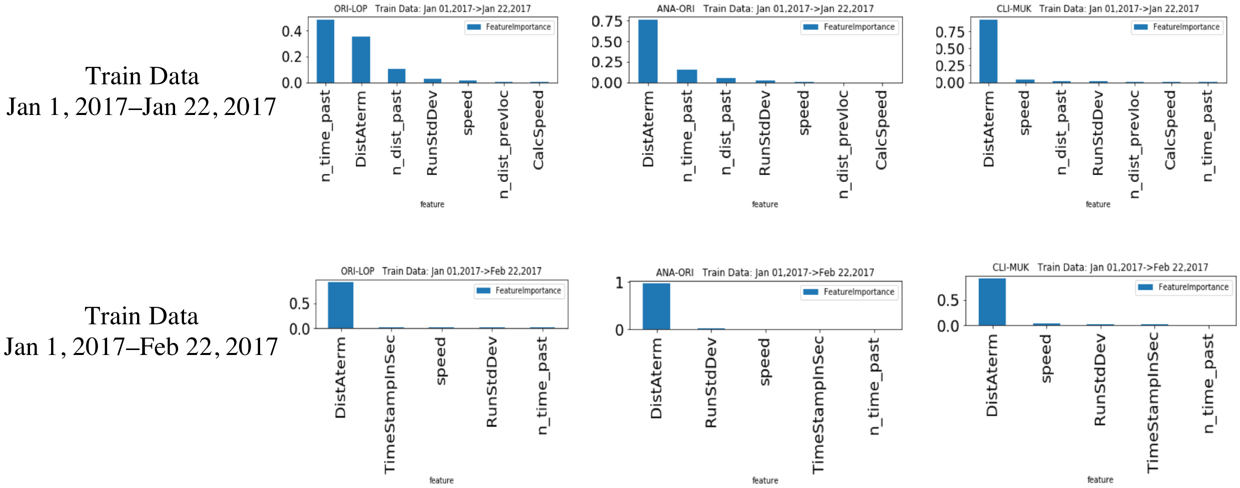


Figure 2: Same route reverse trips have different features

The `feature_importances_` tool could be useful for defining the informative variables for each route and use these variables in future predictions. However, when the training data selected dates change, sometimes variables for `feature_importances_` change too (Figure 3). Because of its long processing time, it might be more effective to find optimum variables to fit the model than using Random Forest `feature_importances_`.


Train Data
Jan 1, 2017–Jan 22, 2017

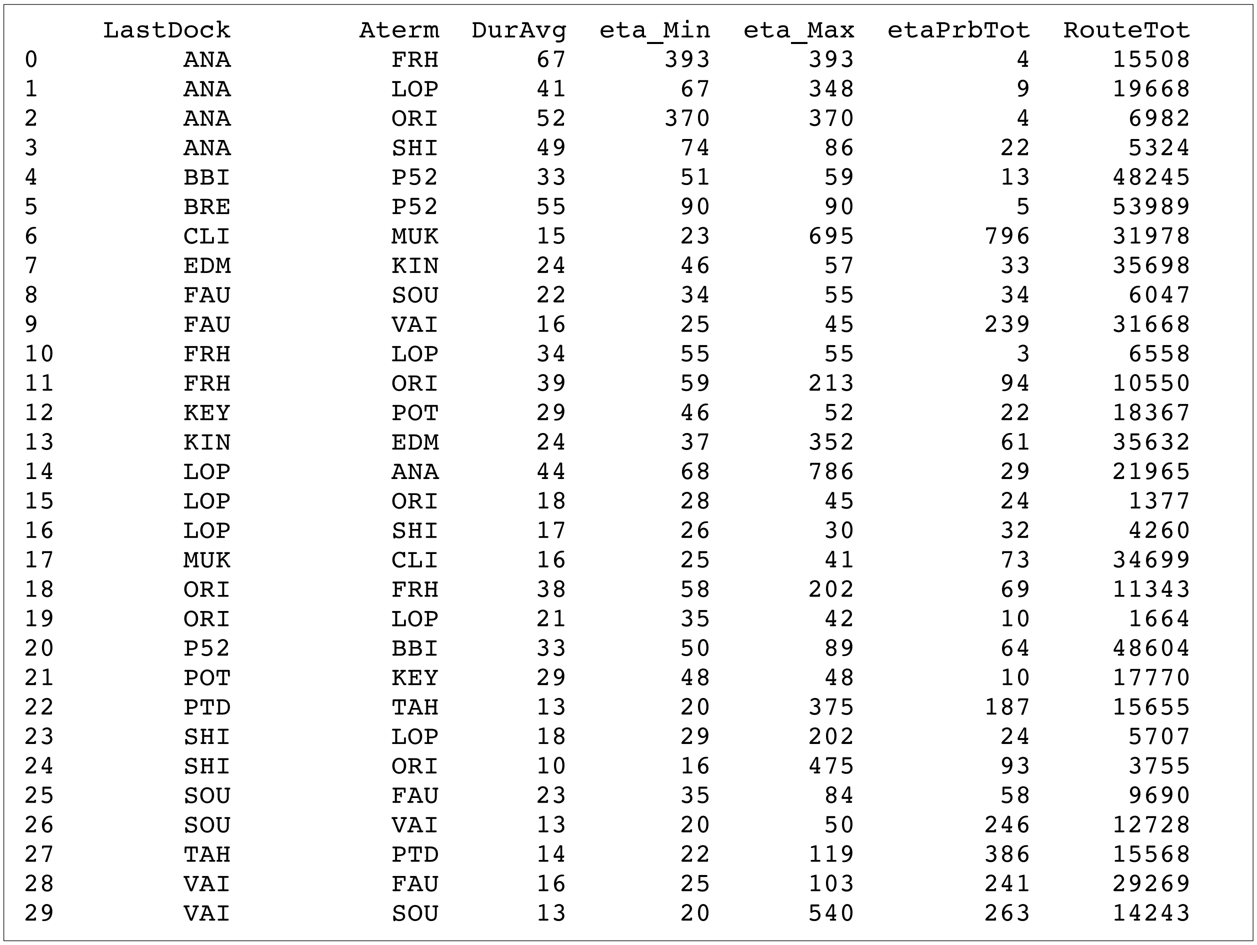
Train Data
Jan 1, 2017–Feb 22, 2017

Figure 3:
If different dates selected for training data, `feature_importances_` variables changes.

5.3 Miscalculated ETA in WSF Dataset

There were some major miscalculations on the WSF file for `eta` variable (Table 3) which was affecting the route ETA accuracy. Washington State Ferry make ETA arrival estimate by using ferries' current coordinate and finding the any previous ferry that had been on the same coordinate. We have this information from `etaBasis` variable in the file for each record. This `etaBasis` variable shows us the `datetime` of the record WSF used to make ETA arrival estimate.

In order to find the reason behind the miscalculations, we did research on this `datetime` records. We found out that, this `datetime` records have some serious problems like those records do not belong to real scheduled working ferries. When we check those `datetime` records, we saw that, these records telling us the ferry on their record is in service (`inservice=` True), but records did not have any `route` information, or `lastdock`, or `aterm` arrival dock information. It means that, the ferries for those records are stopped, mostly waiting in the same coordinates and in reality are not in service. Those `datetime` problem records do not belong to a real trip. When WSF selects those problem records for ETA estimate calculation, they come up with a very bad ETA estimate.



```
    LastDock        Aterm  DurAvg  eta_Min  eta_Max  etaPrbTot  RouteTot
0        ANA          FRH      67      393      393          4     15508
1        ANA          LOP      41       67      348          9     19668
2        ANA          ORI      52      370      370          4      6982
3        ANA          SHI      49       74       86         22      5324
4        BBI          P52      33       51       59         13     48245
5        BRE          P52      55       90       90          5     53989
6        CLI          MUK      15       23      695        796     31978
7        EDM          KIN      24       46       57         33     35698
8        FAU          SOU      22       34       55         34      6047
9        FAU          VAI      16       25       45        239     31668
10       FRH          LOP      34       55       55          3      6558
11       FRH          ORI      39       59      213         94     10550
12       KEY          POT      29       46       52         22     18367
13       KIN          EDM      24       37      352         61     35632
14       LOP          ANA      44       68      786         29     21965
15       LOP          ORI      18       28       45         24      1377
16       LOP          SHI      17       26       30         32      4260
17       MUK          CLI      16       25       41         73     34699
18       ORI          FRH      38       58      202         69     11343
19       ORI          LOP      21       35       42         10      1664
20       P52          BBI      33       50       89         64     48604
21       POT          KEY      29       48       48         10     17770
22       PTD          TAH      13       20      375        187     15655
23       SHI          LOP      18       29      202         24      5707
24       SHI          ORI      10       16      475         93      3755
25       SOU          FAU      23       35       84         58      9690
26       SOU          VAI      13       20       50        246     12728
27       TAH          PTD      14       22      119        386     15568
28       VAI          FAU      16       25      103        241     29269
29       VAI          SOU      13       20      540        263     14243
```

Table 3: List of miscalculated ETA predictions by route

5.4 Trip Base ETA, Predict and Real Data Comparison

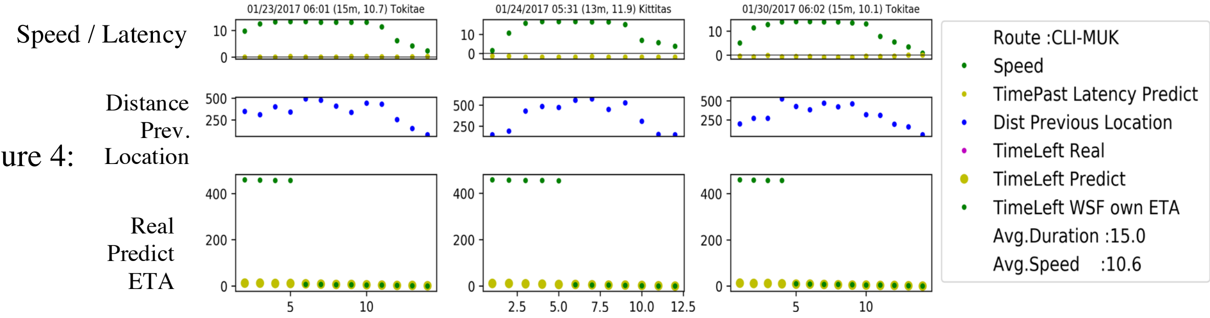
After running all different kinds of regressions, to make an effective analysis and comparison for each route, we visualized the ETA, Random Forest prediction and real `LeftTime` in the same plot. This helped us visually compare our prediction with the real data and WSF estimate ETA by trip base.

Following are some examples of specific characteristics of routes. In those visualizations, there are three graphs for each trip (Figure 4-9). The top graph in the figure shows the speed of the ferry and the latency. It compares trip duration with average and shows us how many minutes the ferry will be late or early. The middle graph shows how much a ferry has traveled from its previous location. The bottom part shows the comparison between WSF's Estimate Time of Arrival, our prediction by using Random Forest, and real arrival time. From those visualizations, we can see the characteristics of each trip, and how close the ETA and Random Forest prediction behaviors compare to real time.

CLI-MUK Route:

The largest difference in ETA miscalculation happens in this route, especially in the first 4-5 minutes of each trip, ETA estimate is very different, but then it catches the real value. Sample graphs of 01/23/2017 @6:01, 01/24/2017 @5:31and 01/30/2017 @6:02 are below (Figure 4):

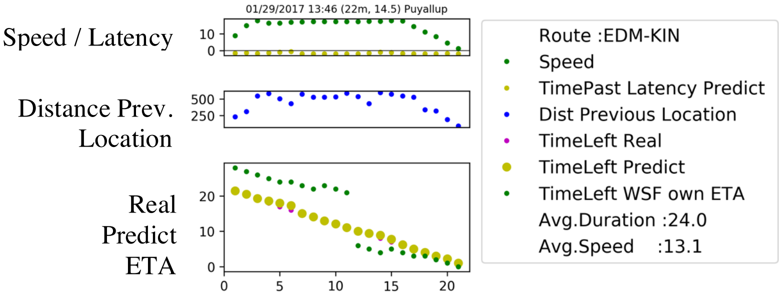
Figure 4:



EDM-KIN Route:

The graph of 01/29/2017 @13:46 is a good example where we see Random Forest prediction is better than ETA prediction (Figure 5).

Figure 5:



FAU-SOU Route:

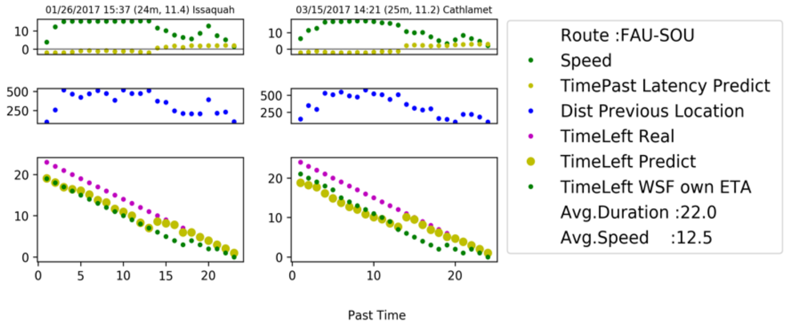
In this graph of 01/26/2017 @15:37, we can see that Random Forest and ETA predictions both differ at the beginning but, Random Forest predicts real left time earlier than than ETA prediction. 03/15/2017 @14:21 is also another example of this pattern (Figure 6).

Figure 6:

Speed / Latency

Distance Prev. Location

Real Predict ETA



FRH-ORI Route:

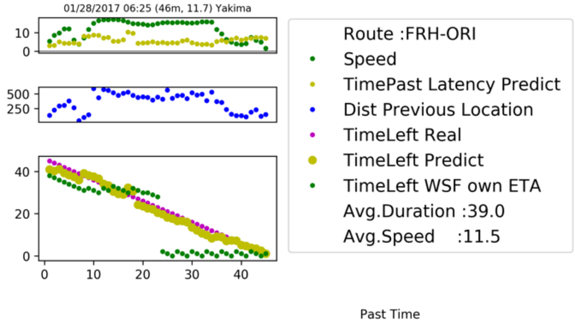
The graph of 01/28/2017 @6:25 is another good visualization for WSF own ETA miscalculation. As we can clearly see ETA wrongly predicted almost entire trip (Figure 7).

Figure 7:

Speed / Latency

Distance Prev. Location

Real Predict ETA



ANA-FRH Route:

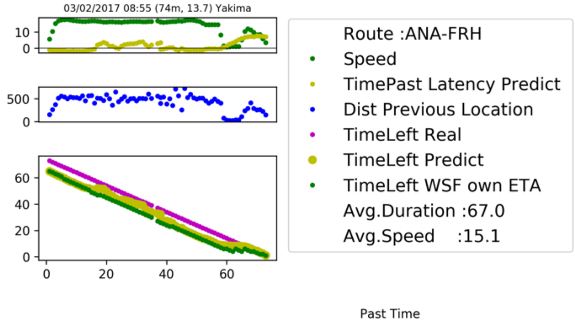
The graph shows that the trip happened on this route on 03/02/2017 @8:55, we can see when the difference in the trip duration happened. Around the end of the trip the ferry slowed down, maybe stopped, and probably waited just before the port for the departure of another vessel. These kinds of graphs can visually show us the possible time of the delay (Figure 8).

Figure 8:

Speed / Latency

Distance Prev. Location

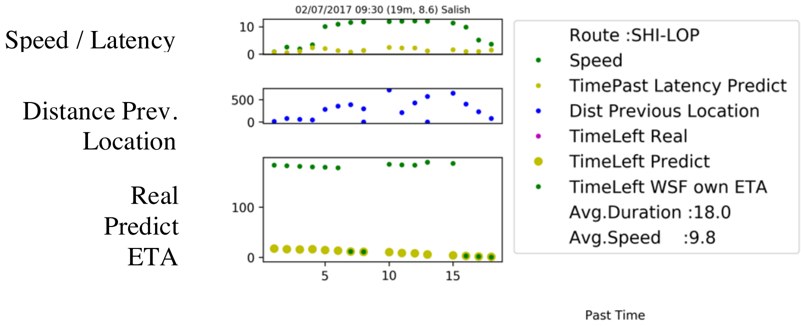
Real Predict ETA



SHI-LOP Route:

In the graph of 02/07/2017 @9:30 we see ETA miscalculation at the beginning, middle, and also at the end. Again, Random Forest prediction is better than ETA prediction (Figure 9).

Figure 9:



5.5 Prediction Analysis

In this section, we analyzed our dataset by route base. To see and to be able to make a comparison with other regressions, we used R^2 score and other error calculations (R^2 coefficient and determination and errors calculations mentioned on Section 4.3). The scores were calculated by using predicted `LeftTime` for each record on each trip and by comparing with real data to show how well the prediction was made. Same score calculations were made for ETA estimate also. In Figure 10, we see the score for three regressions and ETA estimate comparison. The maximum score 1.0 shows the best possible prediction made by the model. In some of the routes, regression scores are all close to each other. However, for the rest of the routes Random Forest's score is significantly better than other two.

There are some odd scores in Figure 10 for WSF own ETA estimates. Some routes have minus WSF own ETA scores which you can see in the numbers from Table 2. In our research, we found that especially for regular trips, WSF own ETA estimates are closer to the Random Forest predictions, but at the same time there are some trips which have huge miscalculated ETA estimates. These bad ETA estimates cause minus scores. The reason behind this miscalculation has been found and explained in Section 5.3.

As we can see from Table 1, the results of all three-regression analysis are close to each other. However, Random Forest has better scores overall, compared to K-Neighbors and Linear Regression. Random Forest is a very effective regression analysis tool to predict the `LeftTime` for the trips. The disadvantage of Random Forest is, when it comes up with a bad trip, the algorithm makes the prediction according to the Training Data minimum or maximum value. This is because there is not similar data in the training data. It is not possible for an average to be higher or lower than training data leaves. Whereas, at Linear Regression, whatever the data is, there will be a result according to the algorithm generated linear equation. During our work we realized that Linear Regression method is faster than other two regression models. However, Random Forest still makes better predictions than Linear Regression.

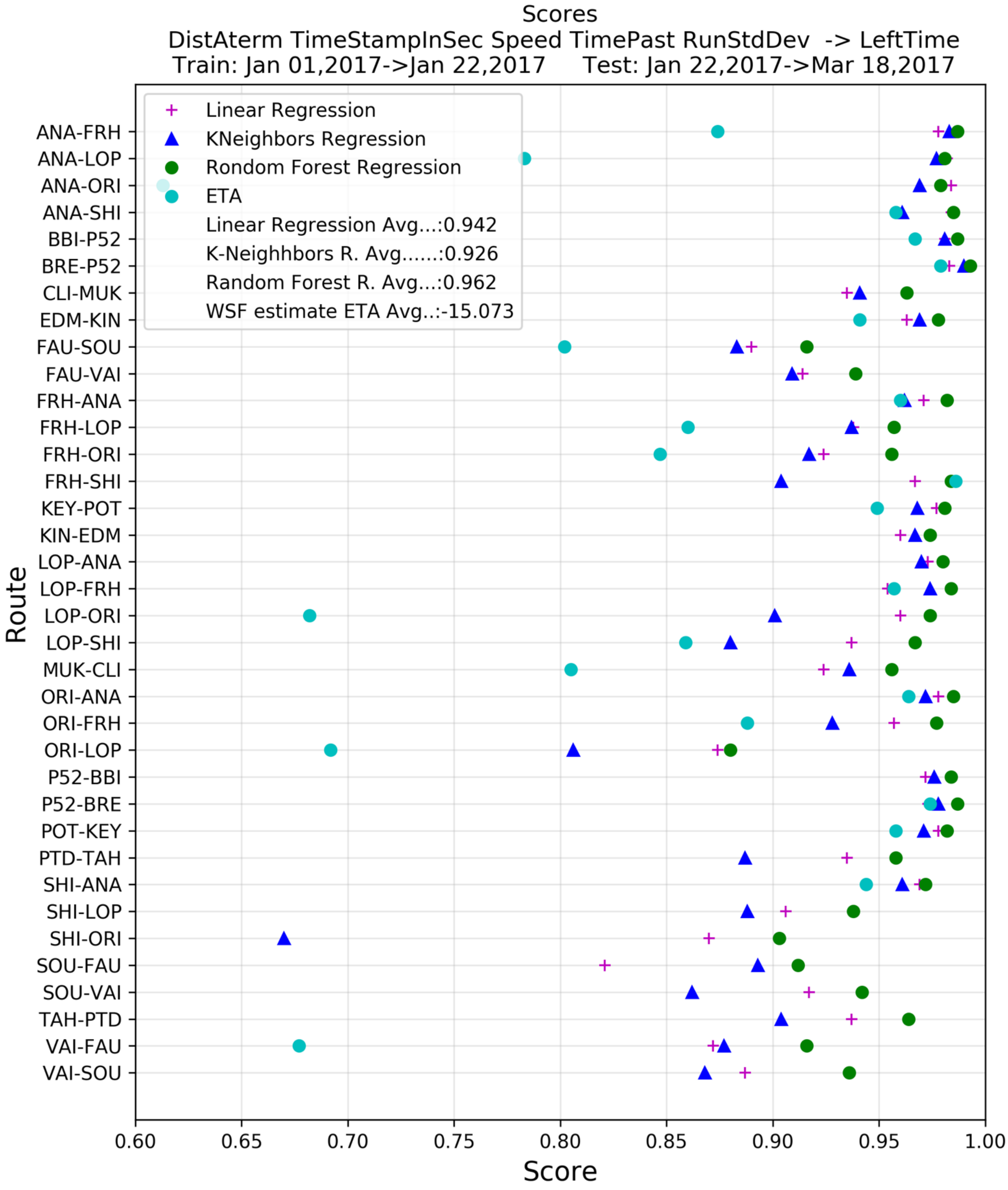

Scores
DistAterm TimeStampInSec Speed TimePast RunStdDev -> LeftTime
Train: Jan 01,2017->Jan 22,2017 Test: Jan 22,2017->Mar 18,2017
+ Linear Regression
▲ KNeighbors Regression
● Rondom Forest Regression
● ETA
Linear Regression Avg...:0.942
K-Neighhbors R. Avg......:0.926
Random Forest R. Avg...:0.962
WSF estimate ETA Avg..:-15.073
Route
ANA-FRH
ANA-LOP
ANA-ORI
ANA-SHI
BBI-P52
BRE-P52
CLI-MUK
EDM-KIN
FAU-SOU
FAU-VAI
FRH-ANA
FRH-LOP
FRH-ORI
FRH-SHI
KEY-POT
KIN-EDM
LOP-ANA
LOP-FRH
LOP-ORI
LOP-SHI
MUK-CLI
ORI-ANA
ORI-FRH
ORI-LOP
P52-BBI
P52-BRE
POT-KEY
PTD-TAH
SHI-ANA
SHI-LOP
SHI-ORI
SOU-FAU
SOU-VAI
TAH-PTD
VAI-FAU
VAI-SOU
Score
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00

Figure 10

5.6 Real Time Data Prediction

After we completed our analysis and decided on our model, we decided to use our model in the real time ETA predictions. We retrieved real time JSON files from WSDOT web site by one-minute interval. For each route, we fitted 2017 February data as a training data to the regression model and predict `n_left_time` from the real-time data. Our aim was to make better predictions than WSF own ETA.

When we compared our predicted results with WSF own ETA estimates, most of the time they had similar realistic arrival time, yet the accuracy of ETA is questionable because of the miscalculated ETA in WSF data set. The reason why ETA estimate is unreliable is due to the fact that WSF uses a single record which is the previous closest location to make ETA estimate. For example, if previous trip is a bad trip or if ferry is not in service then it should not be in the record, yet WSF Dataset showed that these trips existed (see Table 3.)

6 Discussion and Future Work

In this project, we were able to find some errors on WSF own ETA calculation, and saw that this usually happens for short routes and most of the time at the beginning of the trip. The reasons behind this is WSF uses previous closest coordinate location for current location ETA estimate. This selected base closest coordinate location record looks like it belongs to an in-service ferry but in fact it is not. It most-likely belongs to a stopped ferry.

Furthermore, our aim was to find a model to predict the arrival time for ferries better than the WSF's own model for ETA. For this, we used variety of Scikit-learn regressions and tried to see which one works better on our data. Our project shows that the model we made by using Random Forest Regression Analysis can predict arrival time better than WSF own ETA (Figure 10, Table 2), however its process time is longer than others. Although we need to do more studies to draw a conclusion, it would be wise to take this process time into account when choosing the Random Forest Regression model for trajectory calculations.

Based on the results presented in this project, the developed model can improve `LeftTime` ETA predictions. However, during this work the weather factors and ferry traffic were not taken into consideration. So, in the future another study could investigate how would these factors affect prediction results.

7 Conclusion

Our work showed that by factoring in new generated variables like distance to aterm (`DistAterm`), running standard deviation (`RunStdDev`), and utilizing the many features like training data and sklearn tool `feature_importance_` of machine learning algorithms we improved the work outcome and achieved the most accurate predictions of ETA.

The major contribution of this project is to be able to find the errors and the results of these errors on WSF own ETA calculations.

To conclude, machine learning applications are very good in predicting data, and there is always room for better and more accurate results. The results obtained from this work, with further analysis, may help WSF to discover possible reasons for delays, and to improve their services.

8 Github Link

The code has been uploaded on the github, you can access the code on the given link https://github.com/mtduman/Ferry_ETA_Prediction/blob/master/FERRY_Mehmet_D.pdf

9 References

1. Brownlee, J. (2016, June 1). *How To Compare Machine Learning Algorithms in Python with scikit-learn - Machine Learning Mastery*. Retrieved from http://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/

2. Greenfield Advisors. (2014, December 11). *The Love-Hate Relationship Between Property Owners and Passenger Ferries in Puget Sound | Greenfield Advisors*. Retrieved from https://www.greenfieldadvisors.com/the-love-hate-relationship-between-property-owners-and-passenger-ferries-in-puget-sound/

3. Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning*, *30*(2-3), 271-274. Retrieved from http://robotics.stanford.edu/~ronnyk/glossary.html

4. Moseley, D., & Hammond, P. (2011, January). *Washington State Ferries On-Time Performance Report*. Retrieved from http://www.wstc.wa.gov/Meetings/AgendasMinutes/agendas/2011/March22-23/documents/032311_BP17_OnTimePerformanceFinalandappendices.pdf

5. Parolas, I., Tavasszy, L., & Kourounioti, I. (2017). *Prediction of Vessel's Estimated Time of Arrival (ETA) in Container Terminals-A Case Study in the Port of Rotterdam* (No. 17-03164). Retrieved from http://docs.trb.org/prp/17-03164.pdf

6. Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html

7. Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

8. Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

9. Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score

10. Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-error

11. Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

12- Pedregosa, F. et all (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830. Retrieved from http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

13. Perera, L. P., Oliveira, P., & Soares, C. G. (2012). Maritime traffic monitoring based on vessel detection, tracking, state estimation, and trajectory prediction. *IEEE Transactions on Intelligent Transportation Systems*, *13*(3), 1188-1200. Retrieved from http://ai2-s2-pdfs.s3.amazonaws.com/6edf/4273accf9046591d6bf2b8f5dcc6f94837cf.pdf

14. Shanbhag, P. V. (2016). *Finding Patterns in Ferry Vessel Data* (Unpublished master's thesis). University of Massachusetts Dartmouth, New Bedford, MA.

15. Sinnott, R. W. (1984). Virtues of the Haversine, *Sky and Telescope* 68 (2), 159

16. The SciPy community. (2017, January 16). *numpy.corrcoef — NumPy v1.12 Manual*. Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.corrcoef.html

17. Valsamis, A., Tserpes, K., Zissis, D., Anagnostopoulos, D., & Varvarigou, T. (2016). Employing traditional machine learning algorithms for big data streams analysis: the case of object trajectory prediction. *Journal of Systems and Software*. Retrieved from https://arxiv.org/pdf/1609.00203.pdf

18. Washington State Department of Transportation. (2016, December 7). WSF On-Time Performance. Retrieved from http://www.wsdot.wa.gov/NR/rdonlyres/F55B6EFD-3803-47B6-A44F-01E4B058B516/0/PublicOTPReport_June_2016.pdf

19. Washington State Department of Transportation. (2017, June 5). *Vessels Real-Time information*. Retrieved June 5, 2017, from http://www.wsdot.com/ferries/vesselwatch/Vessels.ashx

20. Washington State Department of Transportation. (2017, June 5). *WSDOT - Ferries – VesselWatch*. Retrieved from http://www.wsdot.com/ferries/vesselwatch/Default.aspx

21. Washington State Ferries. (2003). *Washington State Ferries Progress Report July 1, 2001 – June 30, 2003*. Retrieved from Corporate Communications Department Washington State Ferries website: http://www.wsdot.wa.gov/ferries/pdf/progressreport/fullwsfreport.pdf

22. Washington State Ferries. (2017, January). *WSDOT Ferries Division*. Retrieved from http://www.wsdot.wa.gov/NR/rdonlyres/6C78A08B-19A1-4919-B6E6E9EF83E6376D/116404/WSFFactSheet2017_FINAL1.pdf

Appendix 1: Dataset Variables and Generated Variables

Dataset Variables:

vesselID	8	
name	“Elwha”	Vessel Name
lastdock	“Anacortes”	
lasdockID	“1”	
lastdock_abbrev	“ANA”	Last dock code
aterm	“Orcas Island”;	
aterm_id	“15”	
aterm_abbrev	“ORI”	Arrival dock code
eta	“6:20”	ETA Predict
etaAMPM	“AM”	
leftdock	“5:30”	Time of left dock
leftdockAMPM	“AM”	
departDelayed	“N”	
lat	48.596132	
lon	-122.942762	
speed	5.3	
head	334	Vessel Head
datetime	“1/1 06:18”	Vessel Date time
Id	1	Active Vessel Number
inservice	“True”	
system	“WSF”	
mmsi	366773060	Uniquely identify ship stations
route	“ANA-SJ”	
etaBasis	"Vessel Elwha departed Anacortes going to Orcas and using vessel Yakima closest location data from Dec 31 2016 8:17AM"	
nexdep	“5:30”	
nextdepAMPM	“AM”	
lastlat	0	
lastlon	0	
headtxt	“NNW”	
old	0	
timestamp	"1/1/2017 6:19:02 AM"	Time of the file generated.

WSF web site visualization variables:

icon	…green1_335.png"
h	“20”
w	“14”
xOffSet	0
yOffSet	0
pos	“1”
label - icon	“…/366709770.png"
label – h	13
label – w	23
label – xOffSet	0
label – yOffSet	-17
label – place	0

Generated Labels:

n_leftdock	2017-01-01 05:30:00	
n_datetime	2017-01-01 06:18:00	
n_arrive	2017-01-01 06:20:00	
n_time_past	49	
n_time_trip	51	Trip Duration
n_time_left	2	Time Left
p_lat	48.596132	
p_lon	-122.942762	
n_dist_prevloc	0	
n_dist_past	2349.5	
n_distTot	23613.5	
TimeStampInSec	22740	
DistAterm	143.99	
DistTripLastLocTOAterm	30.96	
AT_lat	48.597333	
AT_lon	-122.943494	
p_time_past	49	
p_TimeStampInSec	22740	
TripAvg	52.321	Duration Avg.
Latency	-1.32054	Current Latency
n_Count	96	Trip Record no
etaArrival	2017-01-01 06:20:00	
etaTripDur	50	
etaTimeLeft	1	
RunStdDev	2.5981	Running Std.Dev
n_SpeedCumAvg	15.8625	Speed Cum.Avg.
n_SpeedTripAvg	15.555	Speeg Trip Avg.
SpeedAvg	15.100	Speed Route Avg.

Appendix 2: Missing Trips

For the trips, distance between last location to the arrival Aterm dock longer than 1000 meters has been removed:

Total problem records	: 14,366	
Explored total records	: 10,000	(See below for map)
Records for other routes	4,366	

Fig 1: 1,050 records' coordinates show route VAI-SOU, but in the file the route is VAI-FAU
Fig 2: 1,310 records' coordinates on file VAI-FAU, but in reality mixed up with SAU-FAU-BBI
Fig 3: 1,069 records on file ORI-SHI, but in reality it's ORI-ANA
Fig 4: 1,376 records in FRH-ORI, but distance last location to aterm is min 1430m, max 8583m
Fig 5: 1,469 records in FRH-ANA, but distance last location to aterm is min 1063m, max 25007m
Fig 6: 1,098 records in ANA-FRH, but distance last location to aterm is min 4463m, max 25108m
Fig 7: 1,287 records in ANA-LOP, but distance last location to aterm is min 2322m, max 90990m
Fig 8: 505 records in FAU-VAI, but distance last location to aterm is min 1026m, max 5196m
Fig 9: 836 records in BBI-P52, but distance last location to aterm is min 1039m, max 13471m

Fig 1:

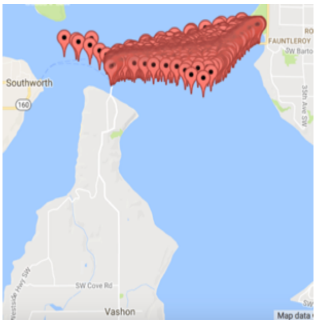


Fig 2:

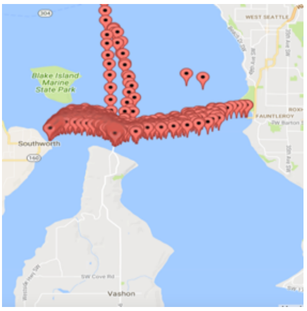


Fig 3:

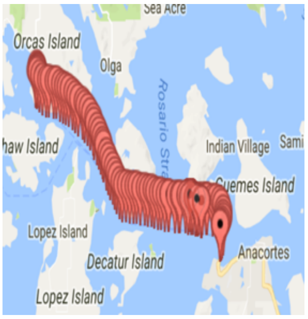


Fig 4:

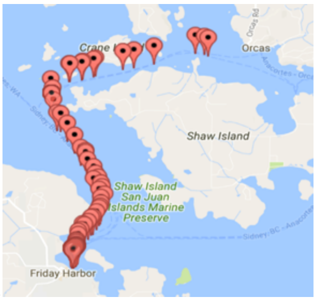


Fig 5:

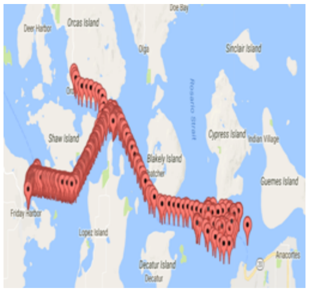


Fig 6:

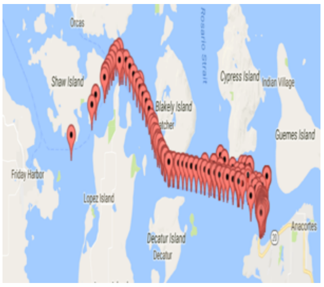


Fig 7:

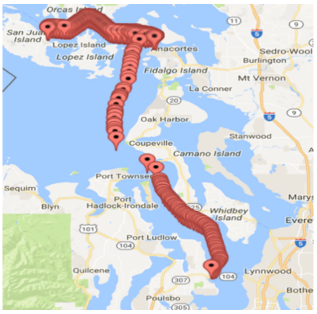


Fig 8:

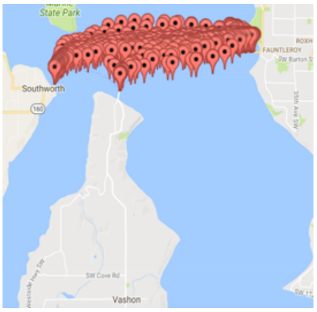
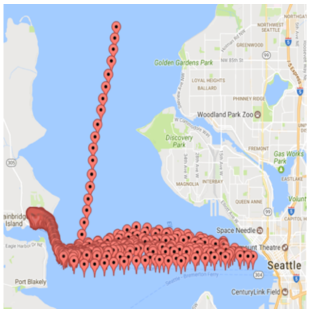
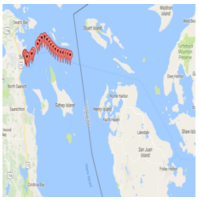


Fig 9:



Appendix 3: Problem Trips

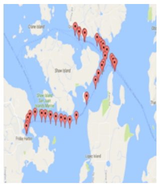
Fig 1: FRH-SID; n_leftdock=2017-01-02 9:55 trip has missing info
Fig 2: ORI-SHI; Vessel Chelan departed Orcas going to Shaw Island route 2 Trips, both belong to a different route in reality.
Fig 3: SID-FRH; n_leftdock = 2017-01-17 12:04 trip has missing info
Fig 4: a) LOP-SHI; n_leftdock = 2017-02-08 21:55 trip has FRH-ORC info
b) LOP-SHI; n_leftdock = 2017-02-18 20:27 trip has LOP-ANA info
c) LOP-SHI; n_leftdock = 2017-02-23 20:24 trip has LOP-ANA info
d) LOP-SHI; n_leftdock = 2017-02-08 16:00 trip has wrong info
e) LOP-SHI; n_leftdock = 2017-02-23 13:53 trip has LOP-ANA info
Fig 5: ORI-LOP; n_leftdock = 2017-02-13 13:25 trip has ORC-ANA info
Fig 6: a) SOU-FAU; n_leftdock = 2017-01-26 01:41 trip has SOU-VAI info
b) SOU-FAU; n_leftdock = 2017-03-12 23:06 trip has SOU-VAI info
c) SOU-FAU; n_leftdock = 2017-01-01 23:06 trip has missing info
d) SOU-FAU; n_leftdock = 2017-01-02 23:07 trip has SOU-VAI info
e) SOU-FAU; n_leftdock = 2017-01-19 00:26 trip has SOU-VAI info
f) SOU-FAU; n_leftdock = 2017-02-02 17:53 trip has SOU-VAI info
g) SOU-FAU; n_leftdock = 2017-03-12 01:41 trip lasted 83 minutes and no GPS data for one hour

1:


2a:

2b:

3:

4a:


4b:

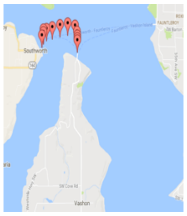
4c:

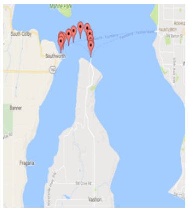
4d:

Orcas Island
Shaw Island
Blakely Island
Lopez Island
Decatur Island

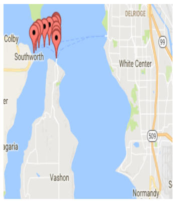
4e:

5:

6a:


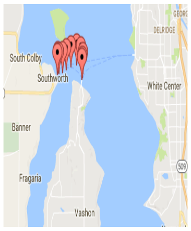
6b:


6c:

6d:


6e:

South Colby
Southworth
White Center
Banner
Fragaria
Vashon

6f:

DELRIDGE
South Colby
Southworth
White Center
Banner
Fragaria
509
Vashon