Temporal Data Mining

Vedashree Bagade, Ankit Gupta, *University of California, Santa Cruz*

03/17/2016

We look at the various techniques and tools used for the analysis of time series data. Time series data is ubiquitous in nature, finance and science. Hence it becomes important to understand how we can extract important information out of the data. In this project, we examine data sets from different domains like automation, health/fitness and stock market. We also study how analysis of a time series data differs from a regular data.

1 Introduction

Time series data is ubiquitous. From change in temperature and pressure over the entire year to detecting gravitational wave signals coming from objects unfathomably far away, many applications require analysis of temporal data in order to make out something useful out of it. In September 2015, scientists at LIGO detected the first direct gravitational waves. But it took them almost 4-5 months to process (or mine) the data in order to detect the signal itself. However, these brilliant scientists were generous enough to open source the code and even provide a tutorial explaining how they mined the data to detect the actual gravitational wave, an event never done before in the history!

This project is inspired by the work done by scientists at LIGO. However, since we are not brilliant enough to work on something as big as gravitational waves, we decided to tone down the project and work on some simple, yet useful, data sets. We examined time series data analysis techniques for data sets from three different domains: finance, health/fitness and automation. We chose data sets from different fields in order to get a more general view of temporal data mining techniques.

2 Temporal Data

2.1 What is temporal data?

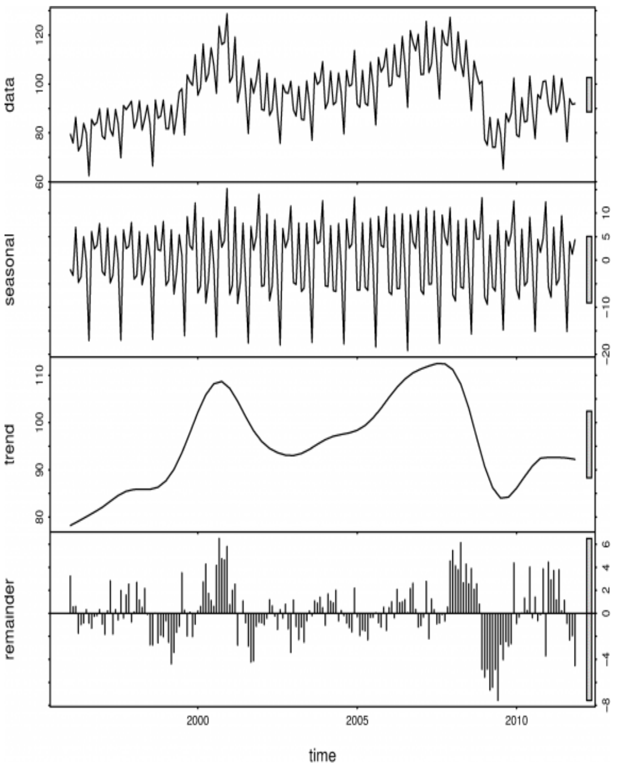
Temporal data is nothing but observations of a feature or a quantity taken at different times. Usually, the data is recorded after a constant time interval which can be minutes, hours, days, weeks, etc. Example: In order to observe the trend or change in temperature over an entire year, we can keep a number of temperature sensors at different places in a city and take a reading from them every 1 hour. This way we will have 24 readings from each sensor for a day and we can use this data to extract information like how temperature varies at different times of the year, etc.

2.2 Components of time series data

A time series data can be split into various components like trend, seasonal and cyclic.

Trend: A trend can be considered as a parameter to see if a given signal increases or decreases for some extended period of time. The trend curve will have a linear curve for such a data. However, most of the times signal might continuously decrease after increasing so our trend curve can also change its direction.

Figure 1: *Example of different components of a time series data*



Seasonal: Seasonality of a time series data is associated with the effect over a certain period of time, like month or a quarter, of year have on the given signal. Example: Sales of umbrellas is higher during monsoon and stays very low during other seasons.

Cyclic: When data increases or decreases without any fixed period, it has a cyclic component. The period of cyclic component is generally more than that of the seasonal component.

Residual: This is not really a component but the remains of the signal. We get the residual component of a time series signal by subtracting the above mentioned components from the remaining signal. So, basically residual signal is not influenced by any trend or seasonal behaviors. Residual signal is very useful in time series data mining.

2.3 Signal decomposition

We can decompose the time series signal into its components by using seasonal decomposition function in python. Once, decomposed we can use each of these components independently or in some desired combination to understand the original signal phenomenon. There are two popular ways of combining these components:

Additive: Y = T + S + R
Add all the signal components, this will basically return the original time series signal

Multiplicative: Y = T * S * R
Multiply all the signal components. The amplitude of this signal can be much higher or lower compared to the original signal.

2.4 Moving Averages (MA)

The moving average method is one of the most famous techniques used in time series data analysis. It is used to estimate the trend component of the signal. In simpler terms, it can be thought of as a smoothing function over the signal. The reason why it is so important is that it neglects any unnecessary noise spikes from the original data. It wont be wrong to say that it basically acts as a shield protecting the data from noise.

The moving average of a function at time t can be calculated using below formula:

$T_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}$

We can observe from the equation above that the trend cycle at time t is calculated by taking mean of the time series within k periods of time t. Observations that are nearby in time are also likely to be close in value, and the average eliminates some of the randomness in the data, leaving a smooth trend-cycle component. This is called an m-MA meaning a moving average of order m.

2.5 Autoregression (AR)

Autoregression performs exactly similar operation over a time series data that linear regression model does on any ordinary instance based data. The autoregression model basically associates a given instance with the past instances as well since in a time series data the instances that are close to each other are more likely to be same.

So, just like linear regression model, the autoregression model also associates a certain weight vector with each attribute. We can specify before running the function if we want the relationship to be linear or polynomial. The autoregression model also returns a constant value which indicates the y - intercept of the line that can be drawn over the signal. The performance of the AR model largely depends on the order we use it with. Some signals perform better at lower order while some might be most efficient at higher order. That is, we can make more predictions using a line for some signal while for some other signals we might need a parabola or some other complex figure to predict future values.

2.6 ARMA

ARMA is an inbuilt function in the statsmodel api in python that can be used to generate the autoregression and moving average coefficients and constants. The value of the AR and MA order can be obtained using the autocorrelation and partial autocorrelation plot for the original signal. However, brute force technique of running ARMA function for various orders of AR and MA and then analyzing the results for the most optimal value is also a way to accomplish this task. In this project, we use the later technique when the former one did not seem to work.

2.7 ACF and PCF

Autocorrelation(ACF) is the correlation between neighboring observations in a time series. When determining if an autocorrelation exists, the original time series is compared to the lagged series. Partial Autocorrelation(PACF) technique is used to compute and plot the partial autocorrelations between the original series and the lags. However, PACF eliminates all linear dependence in the time series beyond the specified lag.The PACF is most useful for identifying the order of an autoregressive model. Specifically, sample partial autocorrelations that are significantly different from 0 indicate lagged terms of a signal value that are useful predictors of its value in future.

3 Dow Jones

3.1 Goal

Based on the stock prices available for the past weeks, we try to predict the price we can expect in the next week. Specifically, we predict the percent change in stock price for the next week based on given attributes.

3.2 The data set

We obtained the Dow Jones data set from UCI Machine Learning website. The data set was generated by Dr. Michael Brown of the University of Maryland, College Park for his research. However, this experiment can be performed by generating data from different sources like Yahoo finance.

The Dow Jones data set consists of weekly stock price changes for 30 different companies listed on the Dow Jones index of USA. We have 25 weeks of data available for 30 companies. Each row in the data set represents weekly data for a certain company.

The data set also has 16 attributes. Before we proceed with the in depth analysis of each attribute to understand which ones are more significant than others, lets take a look at all 16 attributes:

1. quarter: the yearly quarter (1 = Jan-Mar; 2 = Apr=Jun).
2. stock: the stock symbol
3. date: the last business day of the work (this is typically a Friday)
4. open: the price of the stock at the beginning of the week
5. high: the highest price of the stock during the week
6. low: the lowest price of the stock during the week
7. close: the price of the stock at the end of the week

8. volume: the number of shares of stock that traded hands in the week
9. percent_change_price: the percentage change in price throughout the week
10. percent_chagne_volume_over_last_week: the percentage change in the number of shares of stock that traded hands for this week compared to the previous week
11. previous_weeks_volume: the number of shares of stock that traded hands in the previous week
12. next_weeks_open: the opening price of the stock in the following week
13. next_weeks_close: the closing price of the stock in the following week
14. percent_change_next_weeks_price: the percentage change in price of the stock in the following week
15. days_to_next_dividend: the number of days until the next dividend
16. percent_return_next_dividend: the percentage of return on the next dividend

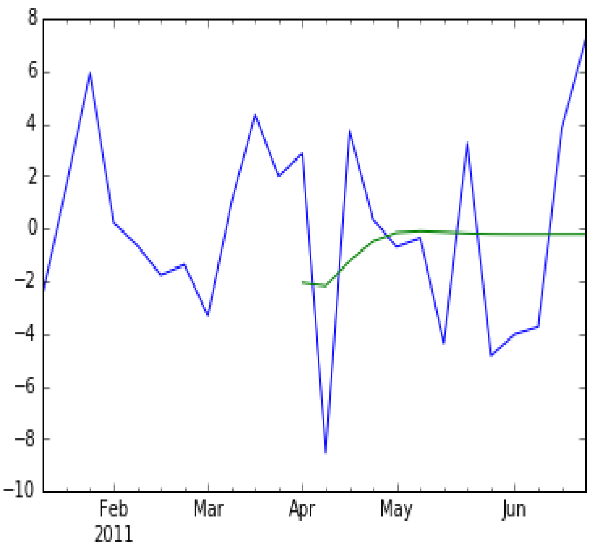
If we observe closely, we can see that attributes like percent_change_price and others are derived from the original attributes like open, high, close. This is a neat technique to compress the feature set since we are mostly interested in if a particular share made a profit or a loss in this week, based on which we predict the next week's performance. Since we reduce the number of features by also maintaining all the necessary information we need, we can expect some performance gain during analysis.

3.3 Analysis

In the beginning, we simply used Linear Regression for predicting the change in a stock's price for the next week based on previous week's data. Surprisingly, we also obtained good results for that, around 70% accuracy. But this approach is wrong. In this approach we do not treat the data as a time series, we treat each instance as an independent individual entity. This is not true for time series data since each instance depends upon its previous instances. To resolve this issue, we decided to approach this problem by using two different techniques.

In the first approach, we calculated the constants and co-efficient vectors using ARMA function by using 24 weeks of data for each company. Based on this we obtained a new data frame of size (30 x 6). This new data frame represents the trend behavior of a particular stock during past 24 weeks. Using this information we can generate a hypothesis to predict future value. We used Linear regression for this purpose. Linear Regression calculated weight vector for all the attributes. These weight vectors are same for all the companies! Now, for testing the performance of this data set, we use the 25th week's data for each company. We first did the same transformation and calculated co-efficients and constants by running ARMA function from 1st week to the 25th week of data for each company. The test data set is again of size (30 x 6). Now we simply use the predict function available in python over the previously trained classifier. Using these predicted values and the actual target values, we calculated an accuracy of about 78%. We also performed the same experiment by training the classifier for data for quarter 1 and then predicting the gain for quarter 2.

Figure 2: *ARMA(2,1) model results for first approach*



The first approach seems to work fine giving a reasonably good accuracy as well. However, it has some limitations. It generates only one classifier for all the companies. Most of the times a company's stock increases or decreases based on that company's performance in recent days and not by the performance of every other company. In the second approach, we generated a separate classifier for each company. Training a classifier for just one company can also provide some in-depth information about that company as well and predictions can be more accurate.

3.4 Results and Conclusions

Now let's visualize the predicted values and compare them with the true values. First we will look at the results of the first approach. In figure 2, blue line is for true stock price and green is for predicted stock price. The ARMA model is applied using 2 and 1 order for AR and MA respectively. This can be represented more generally as ARMA(2,1) model.

We also did various experiments in order to understand the effect of order for AR and MA. In figure 3, 4 and 5 we compare the performance of (4,0), (4,1) and (4,2) models respectively. From these figures we can observe that as we increase the order of MA, the prediction function averages over more number of previous instances and generates a smoother curve. Thus the prediction function will change very slowly and will not perform well in cases of spikes in true signal.

Figure 3: *ARMA(4,0) model for first approach*

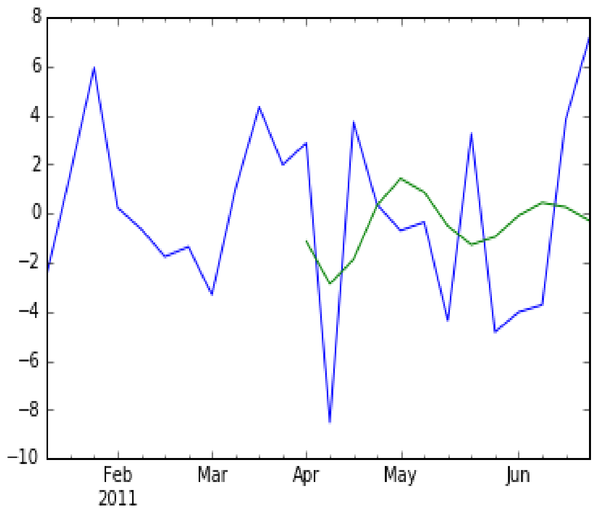


Figure 5 shows the prediction values we generated for second approach and we used ARMA(3,0) model for this.

From the analysis of Dow Jones Dataset we can conclude that the future stock price can depend on a various number of factors which might not be available in the data set. Even though we obtained a fairly good accuracy of about 80%,

Figure 4: *ARMA(4,1) model for first approach*

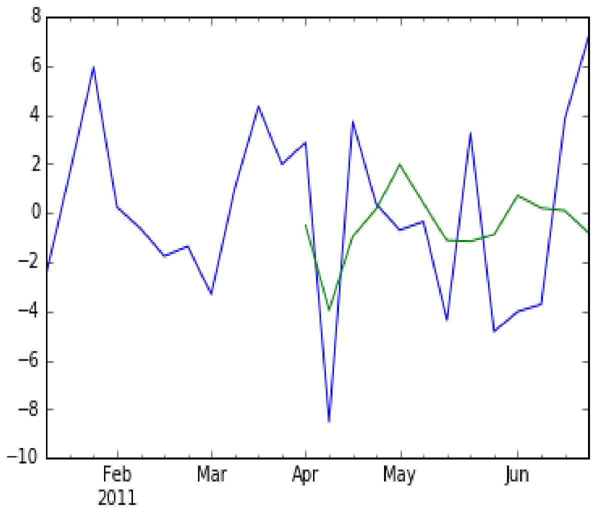
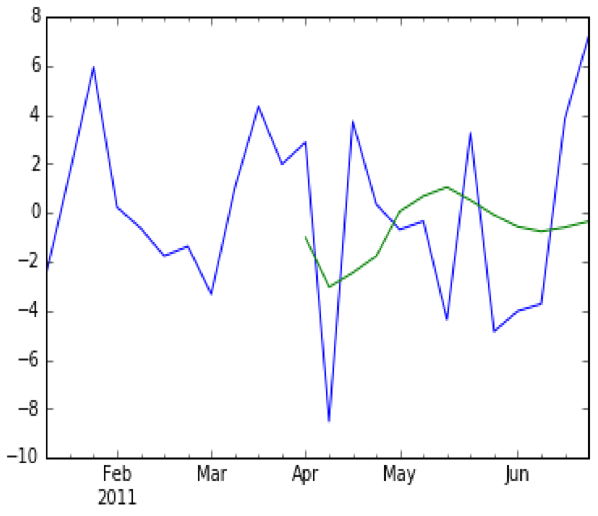


Figure 5: *ARMA(4,2) model for first approach*



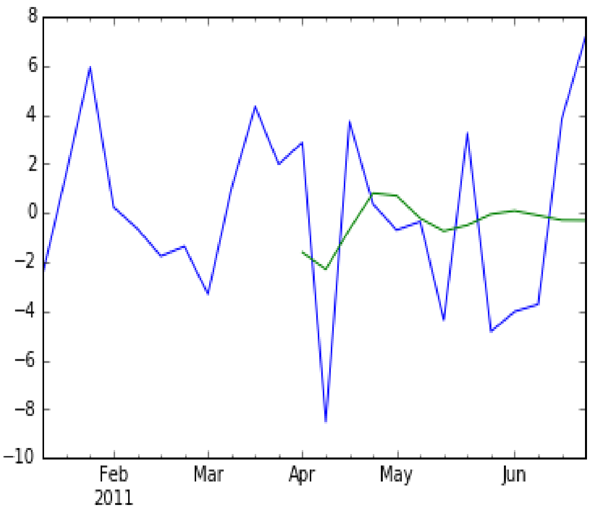
it is not necessary that we will get the same accuracy for any stock prices and for any period of time. It is highly possible that this accuracy we obtained is because of some invisible pattern in our data set. Making a general predictor function for real-time stock prices is a very difficult task.

4 Mhealth

4.1 Goal

Based on sensor reading captured for different subjects while performing different activities like walking,climbing stairs, running, etc, the activity for a subject can be predicted using it values captured for different sensors. This being one application of Mhealth, it can be used in various health related areas dealing with human behavior

Figure 6: *ARMA(3,0) model for second approach*



analysis based on multimodal body sensing.

4.2 The data set

We obtained the data for this experiment from the UCI machine learning repository. The Mhealth dataset comprises of different sensor readings taken for 10 subjects while performing different activities over the duration of 1 min. The readings were taken at a frequency of 50 HZ.

A total of 23 sensor readings comprising:

1. acceleration from the chest sensor, left ankle sensor, right lower arm sensor over X,Y and Z axis
2. gyro from left ankle sensor, right lower arm sensor over X,Y and Z axis
3. magnetomete from left ankle sensor, right lower arm sensor over X,Y and Z axis
4. electrocardiogram signal over two leads

Thus along with 23 attributes the data set includes a total of 12 label classes(representing 12 activities).

The use of multiple sensors captures the body dynamics in a better manner and help in development of the recognition model.

4.3 Analysis

As described above the data set consists of readings taken at a frequency of 50HZ for 1 min. Hence per activity we have a more than 3000 instances for 23 attribute values. But as it is data varying over time each attribute had to considered differently over its 3000 instances and as usual data sets, in this data set a row in the data set cannot be considered as one independent instance. And as its important in time series related data to analyze if there is any underlying trend in the data and also study the autocorrelation between its values for further analysis. We analyzed the trend of the data for some attributes of some labels(activities) for decomposing the time series data into its various components. The trend analysis was done using seasonal decompose module present in python.The ACF and PACF plots were done using the inbuilt functions available in matplotlib in python.

Figure 7 shows the trend analysis performed for acceleration from the left-ankle reading for running activity for 1 subject.

Figure 7: *Mhealth:Trend Analysis for single attribute of one activity*

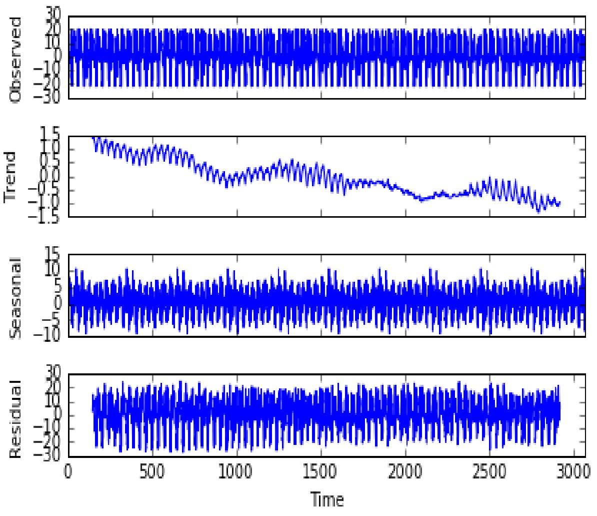
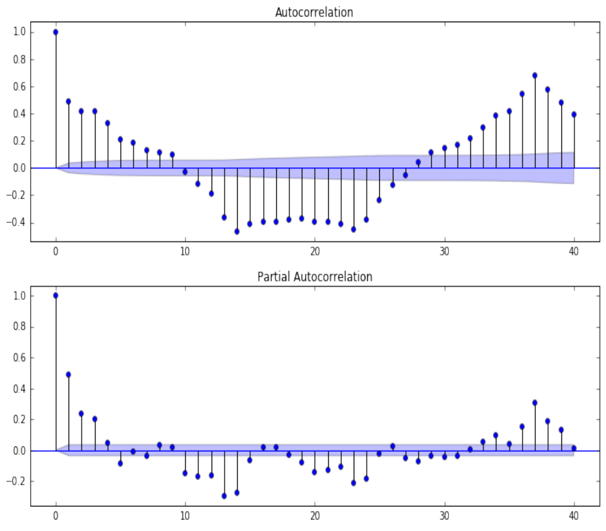


Figure 8 shows the autocorrelation and partial correlation plots for acceleration from the left-ankle reading for running activity for 1 subject.

Although we didn't plot them out here, the other lagged variables show similar results. There could be a relationship to other lag steps that we haven't tried, but it's impractical to test every possible lag value manually. But getting a observation for some of the attributes, that significant lag was observed for 1 lagged terms, we finalized the degree of our ARMA model to be 1.

Hence moving ahead with the analysis of the data set the ARMA(1,0) model was fitted for each attribute for its instances over the entire experiment time duration and so for each label(activity). After fitting the ARMA model for all the 10 subjects, for its 12 activities a total of 120 instances

Figure 8: *Mhealth:ACF and PCF for single attribute of one activity*



were derived. Each instance consists of AR coefficients obtained for each after of its 23 attributes. These coefficients are then passed to classifier for prediction and classification purposes.

To test the performance, we performed classification task using only the coefficients derived by ARMA (23 attributes) as well as using both constants as well as coefficients(46 attributes). In the beginning, we considered only the ARMA derived coefficients assuming the constants won't make difference, but the experimentation later went against it.

Classification was performed using the different data mining tools available in the scikit learn package.

Since this is a multiclass classification problem we carried the classification experiment using:

1. One-Vs-One and One-Vs-The-Rest classifier using logistic regression.
2. K Nearest Neighbors
3. Decision tree classifiers

The performance of all above classifiers was then compared with their accuracies obtained.

4.4 Results and conclusions

Table 1 consists of results obtained from the above tests, i.e. results for different classifiers considering only the coefficients and both the constants as well as coefficients.

As seen from the table there is a considerable improvement in accuracy when considering the both the constants as well as coefficients.

Also we can see from the table that the Decision tree classifier gave the best accuracy for multiclass classification for this data set.

Table 1: *Mhealth Results*

Classifier-	Coeff	Cnsts and coeff
One-Vs-One	67.5	91.66
One-Vs-The-Rest	73.33	87.5
KNN	70.8	92.5
Decision tree	74.16	92.5

5 Robot Execution failure

5.1 Goal

The data set consists of readings captured for different robot related failures and their corresponding force and torque readings. So after a particular failure has occurred for robot the readings captured can be used for prediction of the reason for failures. As robot machines are becoming increasingly popular in the various fields now, analyzing their failure and taking precautionary measures has become important. So data mining the present data for future predictions will be really helpful.

5.2 The data set

The data set consists of 5 failure types, each of them being an independent classification problem by itself.

1. LP1: failures in approach to grasp position
2. LP2: failures in transfer of a part
3. LP3: position of part after a transfer failure
4. LP4: failures in approach to ungrasp position
5. LP5: failures in motion with part

For all of the above five data sets, it includes force and torque readings taken over 3 axis(X,Y and Z) over the period of 315 ms. It includes a total of 15 instances for each label i.e. failure reason.

As these 15 instances taken for each attribute had to be considered as a single case of time varying data for a particular label. We started with analysis of attributes for checking their autocorrelation and partial correlation and well as checking for the trend in the data.

Figure 9 shows the trend analysis performed for acceleration from the left-ankle reading for running activity for 1 subject.

Figure 9: *Robot:Trend Analysis for single attribute of one Failure reason*

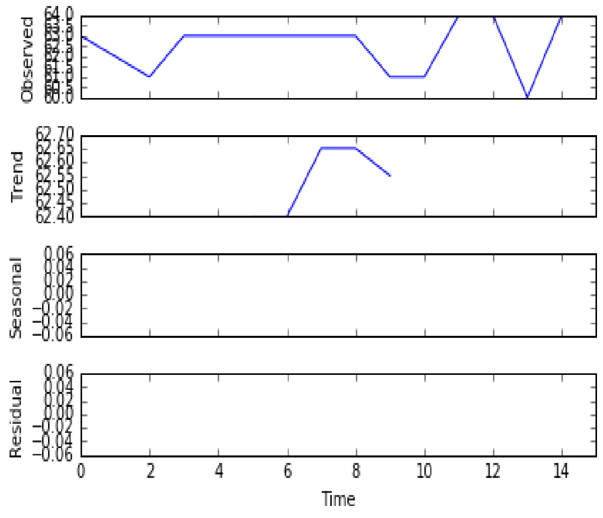
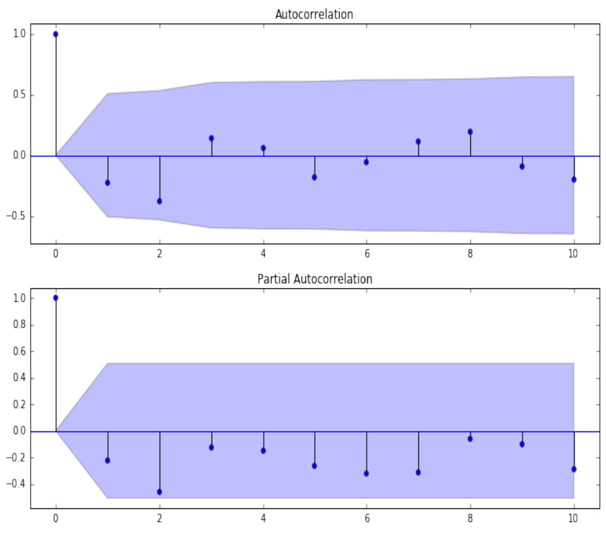


Figure 10 shows the autocorrelation and partial correlation plots for acceleration from the left-ankle reading for running activity for 1 subject.

Figure 10: *Robot:ACF and PCF for single attribute of one Failure reason*



As its not possible to show the plots for all attributes for all failures types, after observing the plots for some of the attributes,we finalized the degree of our ARMA model to be 1 as significant lag was observed for 1 lagged terms,

Hence moving ahead with the above observations on our data, we then imported the data for all the 5 failures types. As Time series are analyzed in order to understand the underlying structure and function that produce the observation here we take the 15 instances captured over 315 ms for 6 attributes for each label and get their ARMA co-efficients and constants. So now instead of (15 * 6)=90 readings per label, we will have 2(constant,co-efficient) * 6 readings per label. As we concluded from our previous experiment that taking both the constants as well as coefficients from the ARMA model results in a better accuracy, we decided to go ahead with that and take both for the experimentation of this data set.

As this data set is also a type of multiclass classification we decided to test our results using different classifiers available in the scikit learn python module and use the similar classification techniques like:

1. One-Vs-One and One-Vs-The-Rest classifier using logistic regression.
2. K Nearest Neighbors
3. Decision tree classifiers

5.3 Results and conclusions

As all the five data sets are independent learning problems. Below are their respective results displaying accuracy in percentage for different classification techniques.

Table 2: *Robot: LP1 Results*

Classifier-	LP1
One-Vs-One	74.73
One-Vs-Rest	65.93
KNN	89.33
Decision tree	91.11

As seen in Table 2, the decision tree gives the best accuracy for LP1: Failure to grasp position data set.

Table 3: *Robot: LP2 Results*

Classifier-	LP2
One-Vs-One	60.33
One-Vs-The-Rest	62.33
KNN	71.2
Decision tree	61.83

As seen in Table 3, the KNN classifier gives the best accuracy for LP2:Failure to transfer part.

Table 4: *Robot: LP3 Results*

Classifier-	LP3
One-Vs-One	55.55
One-Vs-The-Rest	63
KNN	71.5
Decision tree	79.66

As seen in Table 4, the decision tree gives the best accuracy for LP3: Position of part after transfer failure.

Table 5: *Robot: LP4 Results*

Classifier-	LP4
One-Vs-One	89.85
One-Vs-The-Rest	82.99
KNN	95.0
Decision tree	89.09

As seen in Table 5, decision tree gives the best accuracy for LP4: Failure in approach to ungrasp position.

Table 6: *Robot: LP5 Results*

Classifier-	LP5
One-Vs-One	68.7
One-Vs-The-Rest	62.68
KNN	71.0
Decision tree	73.98

As seen in Table 6, the decision tree gives the best accuracy for LP5: Failures in motion with part.

Thus as seen in above tables with results, the maximum overall accuracy of 90 percent is obtained using Decision tree classifier for this Time series, multiclass data set.

6 LIGO: Case Study

6.1 Goal

Involves the first direct detection of gravitational waves. The data set code is open-sourced by the LIGO observatory. The main goal is to understand the preprocessing on strain time series data. As of now there is no learning in this data set as this is the first data set obtained.

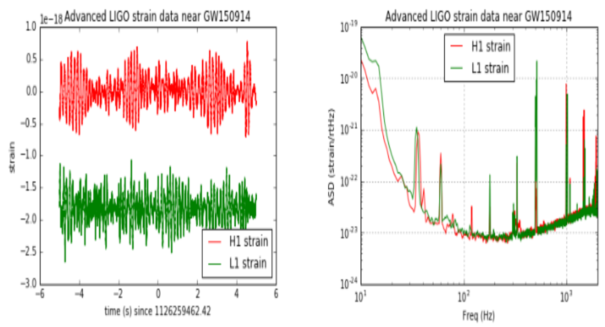
6.2 The Data set

The Data set involves reading from two different sensors(H and L).The data come from the LIGO Hanford Observatory site and the LIGO "H1" detector.Data is sampled at 16384 and 4096 Hertz.

6.3 Analysis and preprocessing the data

The H1 and L1 strain data is shown in figure 11, this is the time series representation of the data. As seen in the figure 11,low frequency noise is dominant in the signal and hence preprocessing the signal to get useful information is very important.

Figure 11: *LIGO: H1 and L1 strain data*



As noise fluctuations are much larger at low, high frequencies and near spectral lines,in the band around 80 to 300 Hz.So whitening the data is always an important step in analyzing such type of data. It involves converting the signal into fourier domain and supressing the extra noise signals, to see the weak signals in the most sensitive frequency bands.

Figure 12 shows the signal produced after whitening.

Another important step involved in preprocessing time domain data is filtering the signal using band pass filter to get signals in a desired band [40 , 300 Hz] here and removing the noise lines from data.

Figure 13 shows the band passed H1, L1 signals.

Other techniques like up sampling and down sampling the signal were also used by the LIGO team to get appropriately detected signals which can be used for further references.

Figure 12: *LIGO: Signal Whitening*

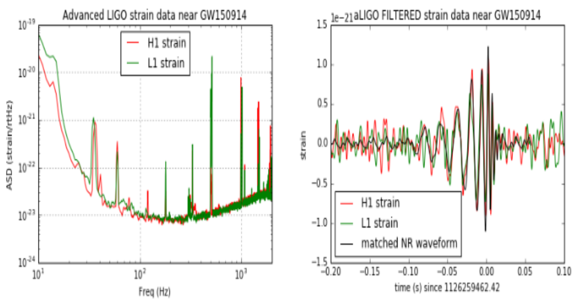
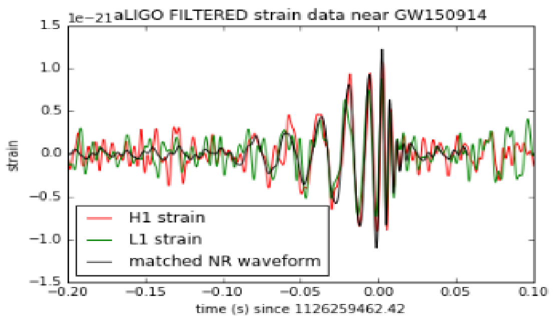


Figure 13: *LIGO: Band Passing*



7 Future Work

For Dow Jones, the stock price in next week can depend on external factors like some recent big announcement from the company which might make people buy more stock of that company and result in greater spike in stock prices compared to what we obtained from just trend analysis. So, we can add a companys repo kind of feature which indicates companys reputation in media based on current event.

For Mhealth and robot, we can integrate the code we have developed with a sensor that provides real time sensor readings. Example: Google Fit, autonomous cars

References

[1] https://www.otexts.org/fpp

[2] http://www.stats.ox.ac.uk/ burke/Autocorrelation/Time20Series20Graphs.pdf

[3] http://www.quantatrisk.com/2014/10/23/garch11-model-in-python/

[4] https://onlinecourses.science.psu.edu/stat510/node/62

[5] http://www.rigtorp.se/2011/01/01/rolling-statistics-numpy.html

[6] http://bicorner.com/2015/11/16/time-series-analysis-using-ipython/