Team 11 – Final Project Report

(Chaithanya Pramodh Kasula and Aishwarya Varala)

Short-Term Energy Consumption Forecasting using Machine Learning

Objective:

Given historical data, the objective is to build a Machine Leaning model to forecast the short-term energy consumption for an individual household.

Dataset Description and Source:

Source: https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption

Description: The current data set titled, 'Individual household electric power consumption Data Set' consists of 2,075,259 rows and 8 columns. Each row represents the active energy consumed every minute (in watt) between December 2006 and November 2010 of a single household. According to the attribute information in the data source, the following are the features that constitute the data set.

Date: Date in format dd/mm/yyyy
Time: time in format hh:mm:ss
Global_active_power: household global minute-averaged active power (in kilowatt)
global_reactive_power: household global minute-averaged reactive power (in kilowatt)
voltage: minute-averaged voltage (in volt)
global_intensity: household global minute-averaged current intensity (in ampere)
sub_metering_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric, but gas powered).
sub_metering_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
sub_metering_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

Task:

The Machine Learning task to be performed is Regression. Given the data, the aim is to predict the values for the features 'voltage', 'global_intensity', 'sub_metering_1', 'sub_metering_2' and 'sub_metering_3' for every day of a week.

System Design and Architecture:

The architecture of the system is shown in Fig. 1. The data source is a csv file that contains the readings of the electric power consumption of an individual household. In order to replicate a real-time distributed system, a streaming **Hadoop architecture** named Kafka is deployed in the project. The Zookeeper of Kafka is the manager which supports synchronization between servers, manages cluster nodes and everything

that is a part of Kafka such as topics, partitions etc. It acts like a controller and is the most important part of the system. A broker can be called as server that runs on the Kafka environment. A producer publishes the information to the subscribers through topics with the help of a broker. A consumer, who is subscribed to a topic, extracts the messages out of it. To simulate a real-world environment, the data which contains all the readings for the columns of the given data set is streamed by the producer to a topic. The consumer extracts the data from the subscribed topic and performs the whole Machine Learning process.

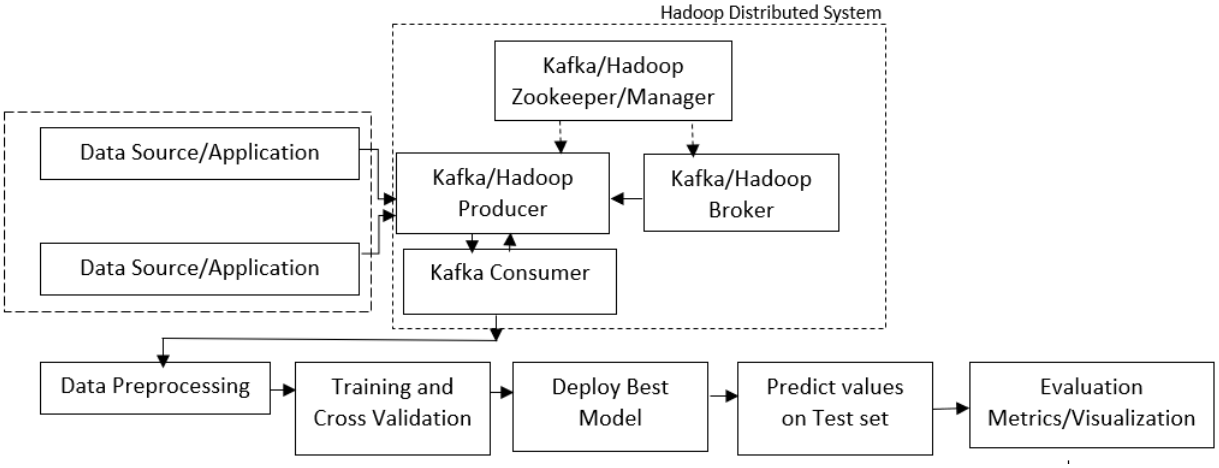

Fig. 1

Preprocessing:

An initial examination of the data shows that nearly 1.2% of the rows in the data have missing values. They are treated using pre-processing techniques. For a given row, if it contains null values, it is replaced by the previous rows' values. This is logical to do so as the readings for the current minute would be very close to the readings in the previous minute.

The data consists of readings collected for every minute. However, the aim of the current task is to predict the values for a given day. Hence, the values of every minute have been aggregated to a single day. This is performed by adding the values of rows for each column belonging to a single day. The size of the data set by day is 1443.

Data Visualization:

Scatterplots have been plotted for all the features. A few outliers have been detected as shown in Fig. 2. Since there are very few outliers, they would not affect the analyzing process and have not been removed from the data set. Also, the over-all energy consumption is observed to be high during the end of the months. The visualization of 'Sub_metering_2' is shown in Fig. 3.

Correlation Plot:

The correlation plot is also drawn for the features in the data set. As. Shown in Fig. 4, 'Global_intensity' and 'Global_active_power' are highly correlated to each other. We do not predict the values for 'Global_active_power' in the current task. Hence, removing the feature does not make a difference.

Data Normalization:

Normalization has been performed for each column in the data set. This is implemented to accommodate all the algorithms being used. Algorithms such as Random Forest are not affected by Normalization. However, other algorithms such as SVM perform better when normalized. Additionally, the RMSE values are better understood when all the features have common results in the range of [0,1].

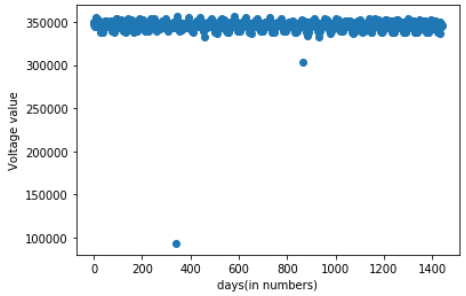

Fig. 2

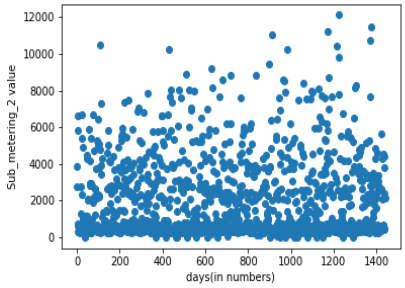

Fig. 3

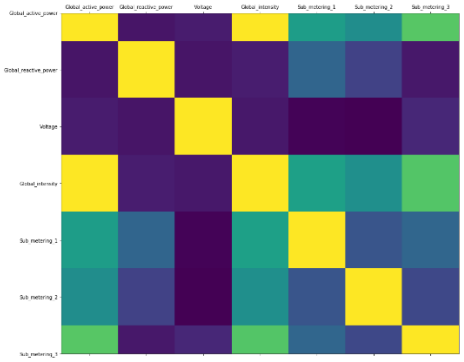

Fig. 4

Model Descriptions and Hyper-parameters:

The current Machine Learning task to be performed is Regression. Random Forest Regression [1], Support Vector Regression [2] and LSTM [3] have been selected for modelling the data.

Random Forest Regression and 0.632 Bootstrap:

Random Forest is an ensemble algorithm. It generates multiple Decision Trees by using a concept called bagging (Bootstrap Aggregation). The trees are created from random subsets of data and their results are not correlated to each other. The chosen number of trees are generated and for each new record, each of the generated trees predict a value. The average of all the predicted values is computed and returned as a result. By assigning Bootstrap equals true, **0.632 bootstrap** method is enabled for the regressor.

Random Forest works well with non-linear data. They reduce over-fitting. They can process data which is in different scales. It is not affected by Normalization. The hyper-parameters used for the project are mentioned below. The process of hyper-parameter tuning is found in '**Hyper-parameter Tuning'** section.

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None, oob_score=False, random_state=1, verbose=0, _start=False)
```

Support Vector Regression:

As mentioned in [4], 'In ε-SVM regression, the set of training data includes predictor variables and observed response values. The goal is to find a function f(x) that deviates from y_n by a value no greater than ε for each training point x, and at the same time is as flat as possible'. The idea is to achieve the minimum possible error by individualizing the maximum margin hyperplane. SVR is not a parametric function. It depends heavily on Kernels. The different types of Kernels are 'rbf', 'poly' and 'sigmoid'. All the three kernels have been tested with the data and 'rbf' has been selected as it results in the least overall RMSE value. The hyper-parameters used for the project are mentioned below. The process of hyper-parameter tuning is found in '**Hyper-parameter Tuning'** section.

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale', kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Deep Learning Framework - LSTM:

LSTM stands for Long-Short Term Memory. It belongs to the family of Recurrent Neural Networks [5]. Traditional Neural Networks are not robust to handle temporal and sequential learning. Whereas, RNNs support such forms of learning. They have loops that which aid in the retention and passage of information from one step to the next. One of the disadvantages of RNN is Vanishing Gradient problem. It cannot process and retain long sequences of information along time. In order overcome this problem, a new architecture named LSTM (Long – Short Term Memory) has been introduced. It consists of three gates which are 'Forget Gate', 'Input Gate/ Update Gate' and 'Output Gate'.

At the 'Forget Gate', a cell decides whether the memory passed from the previous cells must be included in the computation or not. At the 'Input Gate/ Output Gate', the amount of information passed from the previous cells to be included in the current computation. The 'Output Gate' determines the amount of information to be passed to the next cell in the sequence. These gates are represented in Fig. 5 [6]. With the availability of enough data, LSTM's have been proved to work tremendously well. The LSTM architecture used for the project is detailed below.

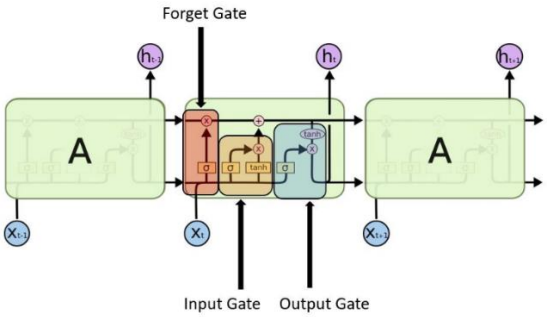

Fig. 5

```
Batch Size = 16
Number of epochs = 5
Number of Neurons in Hidden layer 1 = 200
Number of Neurons in Hidden Layer 2 = 100
Number of Neurons in Last layer = 7
```

Preparing the data for Training and Cross Validation:

Walk forward Validation:

Walk Forward Validation is best suited for time-series data sets. During walk forward validation, the model is initially trained on the whole data set. It is required to predict the first week. Later, the real data for the predicted week is made available to the model and the model is retrained to predict for the second week. This process continues for all the records of the test set. The results for 5-fold cross validation metrics are found in '**5 – fold Cross-Validation**' below.

Recursive Multi-Step Forecast:

Most linear and non-linear models cannot predict values for the whole timeframe. In such cases, recursive multi-step forecast is deployed. For example, instead of predicting all the values for a week at a time, we predict the value for Monday, add this as an input to the model, predict the value for Tuesday and so on. This process is continued until Saturday. Recursive multi-step forecast ensures **high accuracy** predictions.

Performance Metric:

As this is a Regression task, the performance metric used here is Root Mean Squared Error. It is defined as the standard deviation of the errors in predictions. It is a quantification of the spread of data around the best fit line. We calculate the Mean Square Error and perform a square root operation over it to obtain RMSE value.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_i$ denotes the predicted value whereas y_i denotes the actual value. The lesser the RMSE value, the better the model.

Hyper-parameter Tuning:

Hyper-parameter tuning for each model was based on random selection. All the models, except LSTM, have achieved very good performance by selecting the right kernels and by using bootstrap without specifying any other parameters. For LSTM, neurons were added layer by layer to check the right combination to give the best results. Boxplots have been used to assess the error for given number of epochs. The best number of epochs is selected with the minimum variation and minimum error which is 5 epochs as shown in Fig. 6. The activation function used is ReLU as it supports faster training. A dense

layer with 200 neurons forms the first hidden layer. Another dense layer with 100 neurons forms the second hidden layer. The last layer consists of 7 neurons.

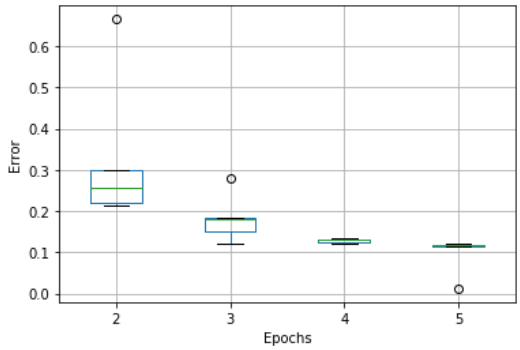

Fig. 6

5 – fold Cross-Validation through Forward Chaining:

As in classification, generic procedure of splitting/shuffling data for cross-validation is not possible with time-series data. This is because shuffling rows disturbs the sequence or the temporal order and results in data leakage. Instead the data is divided into parts without disturbing the sequence. **5-fold cross validation is performed**. For each feature, the RMSE performance during the 5-fold cross-validations are shown in figures 7 to 11.

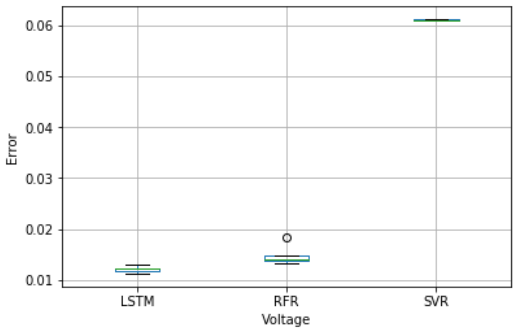

Fig. 7

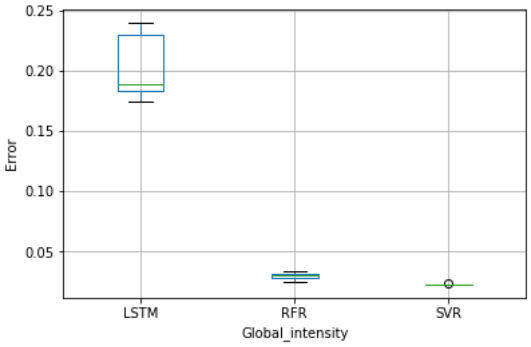

Fig. 8

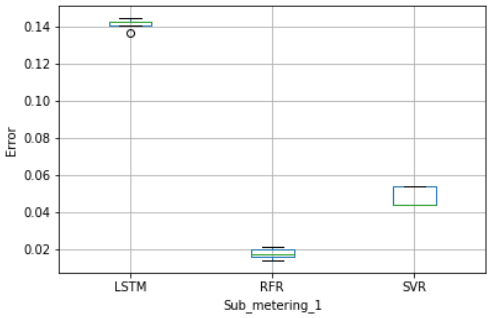

Fig. 9

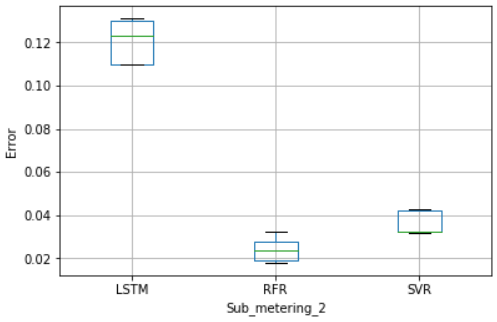

Fig. 10

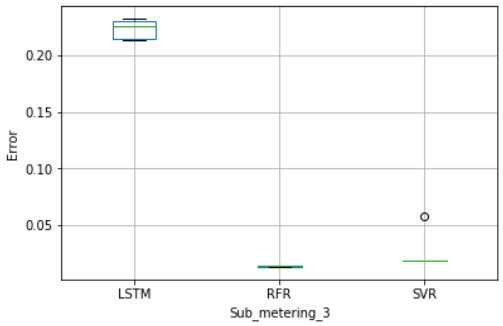

Fig. 11

Model Selection:

The box plots from figures 7 to 11, depict the RMSE values for different models across 5 – fold cross validations. The model must be selected in such a way that the primary importance must be given to the least RMSE value followed by the variation. If the RMSE values for the models are nearly equal, then, the model with the least variation is chosen. Following the same principle, for features 'Voltage', 'Global_intensity', 'Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3', the models 'LSTM', 'SVR', 'RFR', 'RFR' and 'RFR' are selected respectively. Different models for different features are possible as not all models perform better when applied on all the features even if they belong to the same data set. In time series data, when there is a **concept drift**, an ensemble of all models can be used. The 'Performance Evaluation' section also contains plots involving all the models.

Training the selected models with complete Train set (including Freeze and Test):

After cross validation and model selection, the train and the test set are divided in the ratio of 80:20. The train set consists of 1142 records whereas the test set consists of 301 records. The input and output sizes are set as 7. Since the cross -validation phase and the hyper-parameter selection have been completed, the Random Forest Regressor, SVM and LSTM are re-trained over the entire existing training set. LSTM requires the input to be in a specific format which is [records, timesteps, columns]. The trained models are deployed for testing.

Testing:

The trained model is used to perform testing through walk forward validation process. A multi-step recursive forecast is used to predict the test instances as stated above. The results are explained below.

Performance Evaluation:

With the models in consideration, the below figures display the day wise average RMSE values for the test set. With respect to single model selection, for each feature, the overall RMSE values are shown below.

RFR: 0.01911760656, which is the Average test set RMSE scores for features 'Sub_metering_1', 'Sub_metering_2' and 'Sub_metering_3'.

SVR: 0.022569886453231578, which is test set RMSE for the model SVR over 'Global_intensity' feature.

LSTM: 0.01113446515422513, which is the test set RMSE for the model LSTM over 'Voltage' feature.

In relation to the ensemble concept, the plots below include the performance of all the models over all the features. The values plotted in figures 12 to 16 are represented in table 1.

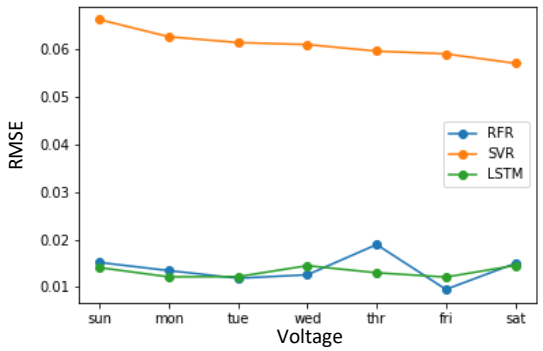

Fig. 12

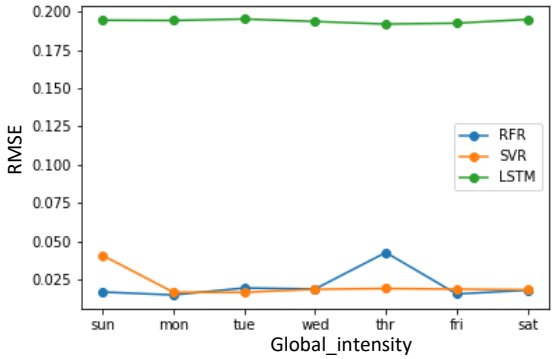

Fig. 13

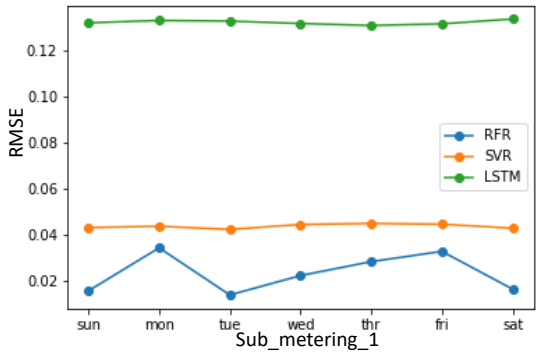

Fig. 14

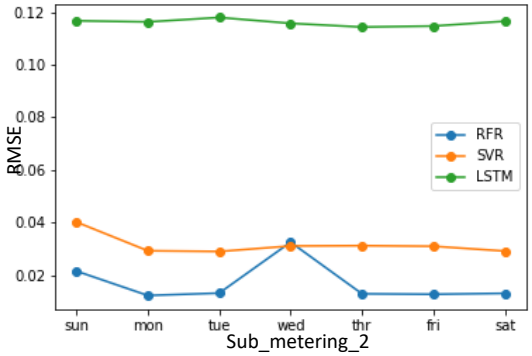

Fig. 15

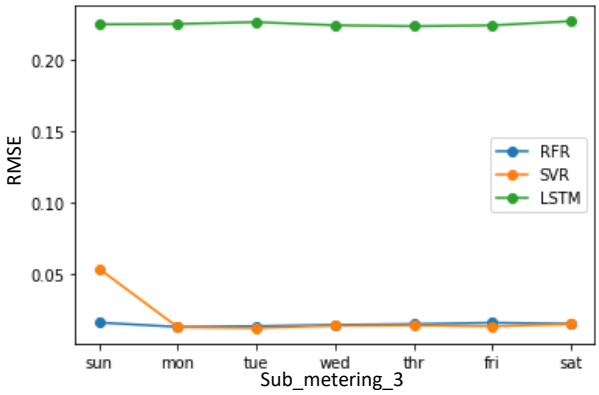

Fig. 16

Average RMSE Values by feature			
	RFR	**SVR**	**LSTM**
Voltage	0.013707694451809646	0.016907405827297373	0.01113446515422513
Global Intensity	0.02260404677164001	0.022569886453231578	0.20337612894998616
Sub_metering_1	0.02461024369303693	0.04369135212097879	0.137077533134143
Sub_metering_2	0.01830478527026438	0.03175787750312785	0.13136172682738684
Sub_metering_3	0.014437790733494106	0.02361861283381553	0.22485160816648692

Table. 1

Discussion and Conclusion:

It can be concluded that Random Forest performs best with features 'Sub_metering_1', 'Sub_metering_2' and 'Sub_metering_3'. SVR performs best with 'Global_intensity' and LSTM performs best with 'Voltage' feature. The models have showcased near perfect results with the best RMSE values. Though LSTM is a powerful architecture, it was not able to work well over the other columns. This can be due to the number of records. With the availability of more training data, LSTM could perform better. However, one of the downsides of LSTM is that it takes a lot of time for training. Random Forest Regression is quick and offers performance very close to that of LSTM. From the cost and time perspective, in the given conditions, we can use Random Forest Regression instead of LSTM. The models are also not overfitted as the train and the test errors for all the models are minimum.

Implementation of tasks that qualify for Extra Credit:

The project includes the implementation of:

(a) distributed data mining using HADOOP – Kafka is used to stream data in Hadoop cluster. The data is stored in HDFS and processed using Map-Reduce concept.

(b) on-line (streaming data) and batch rather than just batch - The Producer-Consumer concept has been implemented in Kafka to simulate real word streaming system. This is integrated with a Machine Learning framework that works on streaming data rather than just batch.

(c) change / data drift (and covariate shift) - Concept drift/ data drift is the phenomenon where there is an occurrence of seasonality. The data set utilized for the current project and the all the features in it are subjected to change with seasons. For example, the power consumption is more in winters than it is in summer and it is expected to change every month with the change in the seasons. As a result, the implemented model is kept up to date by continuously training it with information available every week (most recently available samples) [8] and utilizing an ensemble of classifiers (RFR, LSTM and SVR) [9].

Software and Hardware Discussion:

Hardware: Inter core i7 processor, 8GB DDR4 RAM.

Software: Windows OS, Python 3.6, Anaconda IDE, Keras, Tensorflow, Kafka and other required software/packages.

References:

[1] Sklearn RandomForestRegressor. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[2] Sklearn SVM SVR. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[3] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780.

[4] Understanding Support Vector Machine Regression. (n.d.). Retrieved from https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html

[5] Recurrent neural network. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Recurrent_neural_network

[6] Recurrent Neural Networks and LSTM explained. (2019). Retrieved from

https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9

[7] Concept drift. (2019). Retrieved from https://en.wikipedia.org/wiki/Concept_drift

[8] Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. Machine learning, 23(1), 69-101.

[9] Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks, 22(10), 1517-1531.