# CE 475

# Fundamentals and Applications of

# Machine Learning

# Spring 2018

# Course Project Proposal


**Ege DELİGURU**

**20140601064**

**OUTLINE**

# 1. Abstract

As a project of CE 475 Fundamentals and Applications of Machine Learning, we need to create a program for modeling and understanding complex dataset and then with that model we need to do some prediction for the given dataset. The project is about statistics on computer science and machine learning.

# 2. Introduction

The goal of this project is to predict the Y values for new data points 101 through to 120 as best as I can. I divided the dataset into two pieces such as train and test dataset. In that training dataset, we have 100 data points. The basic definition of that project is that we have a dataset, which has few x values, which are x1, x2, x3, x4, x5 and also we have y, which is an output of these x's values. Our aim is to create a model with these predefined datasets an use it to predict for the dataset which doesn't have Y value in it.

# 3. Identification and Significance of Problem

The project's main topic is to predict 20 data of Y value which are missing. Therefore, I started to do some research on the internet on that topic and I decided to do it in two steps such as classification and prediction.

For the prediction evaluations, firstly I have done some classifications step to finding out which of the data or data are crucial and which are not in order to predict Y values. I applied the methods, which I will explain in detail in the Methodology part. For the training dataset I subtracted the data point which doesn't have y value in it so basically first 100 row become my training data and the rest, which is last 20 become the test data which I will use my model to predict y values in it. Thanks to classification part, I found out which x value is necessary or unnecessary. In this way, I was dealing with less data point while doing prediction and it

was easier for me to think and write my program. Then, I applied a set of prediction methods, which I will explain in detail in the Methodology part again.

# 4. Methodology

First I did the classification part on the training dataset using the Multiple Linear Regression methods which we learned in the laboratory sessions. I heavily used my lab work while doing Multiple Linear Regression to the training set. Then, I have applied loocv method several times while leaving out one x value in each time on the dataset to understand which of the five x values are more important or insignificant with looking their mse value and total mse value. To make it much more clear, the outputs which I get after these methods will be as follows.

I found the order as "x5, x2, x4, x3, and x1". I choose the Multiple Linear regression when I do classification on my dataset because the Multiple Linear Regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y. Then again using the information I learned in the 4. Laboratory, I have done loocv method to understand more about the data and the relationship between them. In addition to that Loocv method, it stands for "leave one out cross validation". It sometimes called rotation estimation or out-of-sample testing is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is the prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run such as training dataset, and a dataset of unknown data as first seen data against which the model is tested that called the validation dataset or testing set. I used this method because it is often claimed that Loocv has higher variance than kk-fold CV and that it is so because the training sets in Loocv have more overlap. This makes the estimates from different folds more dependent than in the kk-fold CV, the reasoning goes and hence increases the

overall variance. I also do a linear regression for the prediction but later I saw it was not the correct way to solve such a problem like that.

As a result of loocv, when I removed the x5 value from the dataset, I realized that the mse value almost never played, that is, almost never affected. But when I removed the x2 or x3 values from the dataset, I noticed that the mse value is affected by an extremely high difference from playing at x5. Therefore, while removing x5 from dataset does not cause depreciation on a method, removal of x2 and x3 values affect the result of the dataset. The examples of my code will be as follows below.

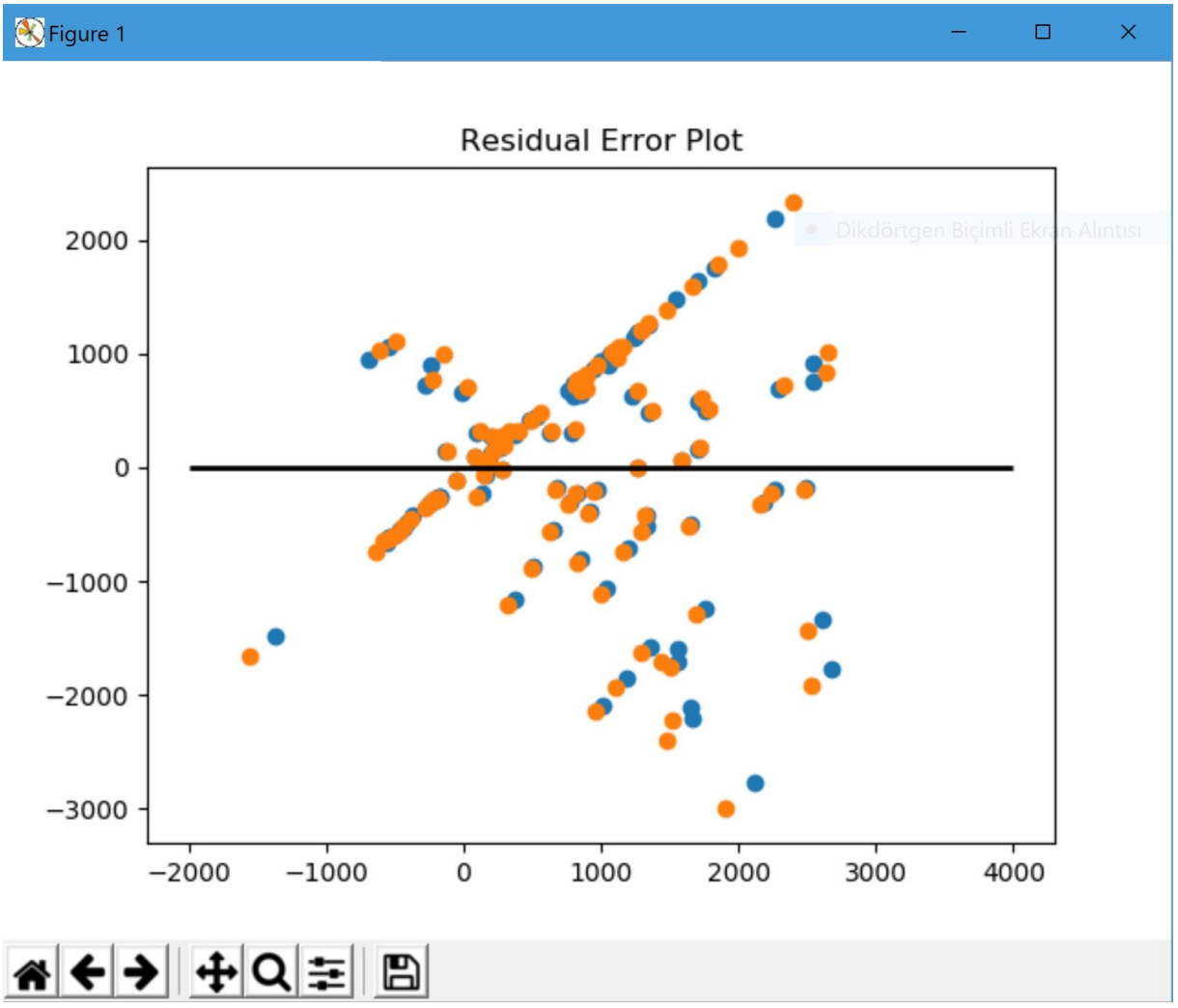
```
import numpy as num
import csv
import math
import matplotlib.pyplot as plt
with open("PDatac.csv") as f:
    csv_list = list(csv.reader(f))
loocv_hat = num.array([])
x1_list = num.array([])
x2_list = num.array([])
x3_list = num.array([])
x4_list = num.array([])
x5_list = num.array([])
y_list = num.array([])
titles = []
titles_sorted = []
for row in csv_list:
    if row != csv_list[0]:
        x1_list = num.append(x1_list, int(row[1]))
        x2_list = num.append(x2_list, int(row[2]))
        x3_list = num.append(x3_list, float(row[3]))
        x4_list = num.append(x4_list, float(row[4]))
        x5_list = num.append(x5_list, float(row[5]))
```

```
        y_list = num.append(y_list, int(row[6]))
    else:
        titles.append(row[1])
        titles.append(row[2])
        titles.append(row[3])
        titles.append(row[4])
        titles.append(row[5])
ones = num.ones((1, len(x1_list)))
mse_loocv = 0

for i in range(len(x1_list)):  # Do the regression for each data point.
    # At each iteration, the i'th point becomes the test data and the rest becomes the train data.
    # We compute each prediction and store them.
    X = num.vstack((ones, x1_list, x2_list, x3_list, x4_list, x5_list)).T
    Y = y_list.T
    X_test = X[i]   # The i'th point becomes the test data.
    Y_test = Y[i]
    X_train = num.delete(X, i, 0)   # The rest becomes the train data.
    Y_train = num.delete(Y, i, 0)
    # Compute the coefficients just like we did in LAB3.
    coefficients = num.linalg.inv(num.dot(X_train.T, X_train))
    coefficients = num.dot(coefficients, X_train.T)
    coefficients = num.dot(coefficients, Y_train)
    Y_hat = num.dot(X_test, coefficients)   # Compute the prediction using the coefficients
    loocv_hat = num.append(loocv_hat, Y_hat)  # Append the prediction into the array "loocv_hat" for
future plotting
    mse_loocv = mse_loocv + math.pow((Y_hat-Y_test), 2)   # Sum the squared errors
mse_loocv = mse_loocv / len(x1_list)   # Average the sum of squared errors
# Computing the regression one last time, as instructed in Task 2
coefficients = num.linalg.inv(num.dot(X.T, X))
coefficients = num.dot(coefficients, X.T)
coefficients = num.dot(coefficients, Y)
Y_hat = num.dot(X, coefficients)
mse_all = num.mean(num.square(Y_hat-Y))
print("MSE with loocv:" + str(mse_loocv))
```

```
print("MSE all:" + str(mse_all))
# Plotting the results
plt.title("Residual Error Plot")
plt.scatter(Y_hat, Y_hat-Y)
plt.scatter(loocv_hat, loocv_hat-Y)
plt.hlines(y=0, xmin=-2000, xmax=4000, linewidth=2)
plt.show()
```

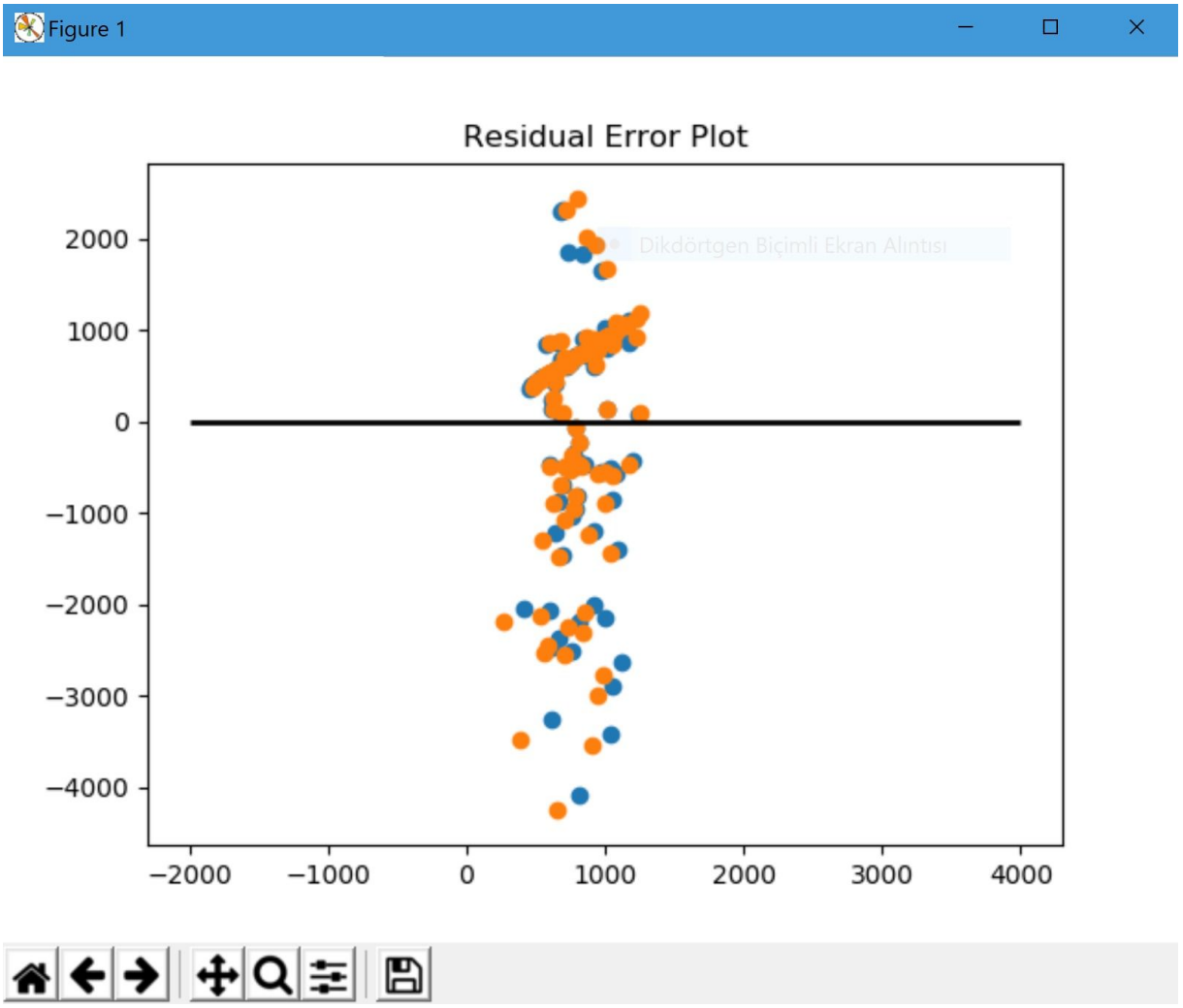
The Graph will be :




The MSE values will be:

```
MSE with loocv:980259.6604973034
MSE all:861404.6290778109
```

But after I delete the x2 and x3 values from vstack I get that graph:




The MSE values will be:

```
MSE with loocv:1714758.1384582594
MSE all:1578164.582034884
```

So from these I understand how x2 and x3 values are important !

After that, I tried to apply feature selection and linear regression methods to dataset. Previouscly, I have mentioned that the values that have the most important factors are as x2 and x3 but with the feature selection I confirmed it again and be sure about that.Code example

```
import pandas as pd
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression


dataset = pd.read_csv('PDatac.csv', encoding='latin1')

X = dataset[['x1', 'x2', 'x3', 'x4', 'x5']]
y = dataset['Y']

model = LogisticRegression()
rfe = RFE(model, 2)
fit = rfe.fit(X, y)
print("Num Features: %s" % (fit.n_features_))
print("Selected Features: %s" % (fit.support_))
print("x1  x2  x3  x4  x5")
print("Feature Ranking: %s" % (fit.ranking_))
```

The output is:

```
Selected Features: [False  True  True False False]
x1  x2  x3  x4  x5
Feature Ranking: [2 1 1 4 3]
```

So with the help of feature selection I also confirmed and get the same result which I got from the multiple linear regression loocv.

I used Future selection method because it enables the machine learning algorithm to train faster. It reduces the complexity of a model and makes it easier to interpret. It improves the accuracy of a model if the right subset is chosen. On the other hand, the linear regression is a single independent variable is used to predict the value of a dependent variable. In multiple linear regression, two or more independent variables are used to predict the value of a dependent variable. The difference between the two is that the number of independent

variables. Also, I have done the linear regression method but I understand that I was on the wrong way to solve the question because it has to have a linear method that is linearly increasing or decreasing. However, our problem is not linear. Later I search for a solution and I founded Keras Library which works on Tensorflow backend.

So finally I found a way to prediction part, I use Tensorflow method to predict 20 Y value. I have tried to use Tensorflow which is an open source machine learning framework for everyone and also I try to use Keras which is a python deep learning library. However, because of not enough experience in machine learning and the Python programming language field, I couldn't able to do my best. Firstly, I install Tensorflow and Keras library than from the documentation from Keras official website and tutorial from YouTube and GitHub relevant issues I try to create a solution for our specific problem. The tensorflow method is an open source library for numerical computation and large-scale machine learning. There are some benefits of using tenserflow, such as it provides for machine learning development is an abstraction. Instead of dealing some details of implementing algorithms, or figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application. Therefore, our problem becomes much easier for me.

I would like to explain in detail with the code and the outputs which I got.Code example:

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.utils import np_utils
from keras.callbacks import EarlyStopping
from keras.regularizers import l1, l2

# Import data
dataframe = pd.read_csv('CE 475 Fall 2018 Project Data - data.csv')
dataset = dataframe.values
```

```
dataframetest = pd.read_csv('CE 475 Fall 2018 Test-data.csv')
datasettest = dataframetest.values
#”CE 475 Fall 2018 Project Data - data.csv” is a csv which has first 100 row that is my training dataset
#”CE 475 Fall 2018 Test-data.csv” is a csv which has last 20 row that is my test dataset

# Random shuffle the dataset
max_num_X = np.amax(dataset[:, 0:-1])
max_num_Y = np.amax(dataset[:, -1])
max_num_X_test = np.amax(20)
max_num_Y_test = np.amax(datasettest[:, -1])

# Split into train and test sets
train_size = 100
#print("ege1", train_size)
test_size = 20
#print("ege2", test_size)
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]


# split labels and datas
def create_dataset(dataset):
    dataX, dataY = [], []
    for i in range(len(dataset)):
        rowX = dataset[i, 0:-1].astype('int')
        dataX.append(rowX)
        dataY.append(dataset[i, -1])
        #print("ege1")
        #print(dataX)
        #print("ege2")
        #print(dataY)
    return dataX, dataY
#Function which split my relevant dataset to the X columns and Y column


trainX, trainY = create_dataset(train)
```

```
testX, testY = create_dataset(test)
#Function call and assigning datasets


# reshape trainX and testX to be [samples, time steps, features]
samples_train = len(trainX)
samples_test = len(testX)
time_steps = len(trainX[1])
trainX_input = np.reshape(trainX, (samples_train, time_steps, 1))
testX_input = np.reshape(testX, (samples_test, time_steps, 1))

# one hot encode the output variable
output_size =  max_num_Y + 1
trainY_cat = np_utils.to_categorical(trainY, output_size)
testY_cat = np_utils.to_categorical(testY,  output_size)

# create the model
input_dimension = 1
model = Sequential()
model.add(LSTM(16, input_shape=(time_steps, input_dimension), dropout_W=0.2, dropout_U=0.2))
model.add(Dense(output_size, activation='softmax', W_regularizer=l2(0.01)))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
print ('Toy LSTM model created...')
#from the official documents

# training
print ('Training...')
early_stopping = EarlyStopping(monitor='val_loss', patience=5000)
model.fit(trainX_input, trainY_cat, nb_epoch=5000, batch_size = 10, verbose=2, validation_split=0.2,
callbacks=[early_stopping])
#Creation of the model

# Save model and weight
model_json = model.to_json()
```

```
with open("Toy_LSTM_model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("Toy_LSTM_model_weights.h5", overwrite= True)
print ("Toy LSTM model and weights saved..")



# testing
print ('Testing on test data...')
scores = model.evaluate(testX_input, testY_cat, verbose=2)
print ('Test accuracy on test data: {}'.format(scores[1]))

print ('Testing on training data...')
scores = model.evaluate(trainX_input, trainY_cat, verbose=2)
print ('Test accuracy on training data: {}'.format(scores[1]))

# Prediction
print ('Predicting on testing data...')
for row in testX:
    x = np.reshape(row, (1, len(row), 1))
    prediction = model.predict(x, verbose=0)
    y_hat = np.argmax(prediction)
    print ((row*max_num_X).astype("int")/max_num_X, "-->" ,y_hat)
#Prediction to for my test dataset

print ('Predicting on training data...')
for row in trainX:
    x = np.reshape(row, (1, len(row), 1))
    prediction = model.predict(x, verbose=0)
    y_hat = np.argmax(prediction)
    print ((row*max_num_X).astype("int")/max_num_X, "-->" ,y_hat)
#Prediction for the training set to see how accurate my code is
```

The output was:

```
Test accuracy on training data: 0.44
Predicting on testing data...
[55. 12.  4. 36.  2.] --> 320
[33. 48. 14. 67.  8.] --> 84
[ 7. 50. 18. 84. 18.] --> 90
[32. 38. -3. 81. -6.] --> 70
[71. 17. 16. 73.  6.] --> 1519
[ 49.  47.  -9.  92. -10.] --> 71
[26. 48.  6. 92.  4.] --> 84
[ 12.  35.  13.  17. -17.] --> 64
[ 27.   1. -10.  83. -12.] --> 208
[26. 29. 18. 15.  7.] --> 1798
[65.  5. 17. 44.  8.] --> 1602
[ 63.  36. -10.  57. -10.] --> 3286
[ 88.  22. -10.  47.   1.] --> 1161
[75. 15.  1. 82. -4.] --> 1312
[18. 28. -3. 59.  2.] --> 89
[93. 39. 18. 21. -3.] --> 70
[ 16.  47.  10.  32. -14.] --> 64
[52. 24. 24. 27.  0.] --> 1602
[ 94.  17.  -6.  42. -16.] --> 4689
[47. 39.  2. 94.  0.] --> 96
```

And also predictions for the relevant rows which you can find in the attached spreadsheet to the document.

The basic explanation is as follow:

Firstly, it will take the first 100 row which is a train data and has all values from x1, x2, x3, x4, x5, and Y. After that with the help of these training data, I try to predict the last 20 data which doesn't have Y values. After adding the required libs, I'm reading the first 100 data, which is the training data of my CSV file. Then I divide them into label and data. To be more specific, I allocate this data as x values and y values. Then I match them to two variables, train x, and train y. I'm doing the same for the data we need to estimate. Then I'm applying the LSTM model on Keras' site to my dataset and recording this model. The result of this model I found the previous step, I apply it to the dataset that the missing 20 Y values on it.

# 5. Implementation

I have written all of my codes on the JetBrains PyCharm environment but ı got some trouble while downloading Tensorflow environment so that I download anaconda in order to follow instruction from the website. When I write the classification methods, I used multiple libraries. Such as, when I applied the Multiple Linear Regression methods to the dataset. I used numpy, csv, sklearn, and matplotlib to import libraries. When I applied the Loocv

method to the dataset, I used numpy, csv and matplotlib to import libraries. When I applied the Feature Selection method to the dataset, I used pandas, RFE and LogisticRegression to import libraries. When I applied the Linear Regression method to the dataset, I used sklearn, pandas, and matplotlib to import libraries. When I applied the Tensorflow method to the dataset, I used Keras libraries such as Sequential, Dense, LSTM, np_utils, EarlyStopping and for other libraries such as pandas and numpy to.

# 6. Result

Since I did not know enough about Tensorflow, I managed to run my code even though I had trouble in at the very beginning however later I solved it and I got 44% accuracy on the training data which is not for me but on the other hand not so good. However, I think it is also related to the training dataset.

# 7. Conclusion

Thanks to this project, I have experienced my very first real-life project in the field of machine learning. While doing that, I used the information which I learned from CE475 classes and I think that I have learned many new topics that might be useful in the future, such as the Keras library and Tensorflow and what'is more I am eager to learn more now.

# 8. References


1) https://www.tensorflow.org/guide/premade_estimators
2) https://towardsdatascience.com/create-a-model-to-predict-house-prices-using-python-d34fe8fad88f
3) http://www.wikizeroo.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTGluZWFyX3JlZ3Jlc3Npb24
4) https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86
5) http://www.wikizeroo.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvRmVhdHVyZV9zZWxlY3Rpb24
6) https://machinelearningmastery.com/an-introduction-to-feature-selection/
7) https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/
8) https://towardsdatascience.com/a-feature-selection-tool-for-machine-learning-in-python-b64dd23710f0
9) https://www.tensorflow.org/tutorials/
10) https://keras.io/layers/recurrent/
11) http://adventuresinmachinelearning.com/keras-lstm-tutorial/
12) https://github.com/topics/multiple-linear-regression
13) https://keras.io/
14) https://www.python.org/