Linear regression and KNN-classification for Weather Data set

Abstract

This is report of the examinations with weather data set. I tried to predict relative humidity based on the weather data set. I used pair plots, PCA, correlation matrix and linear regression to predict relative humidity. Other experiment was to classify the weather conditions either as “dry" or “not dry" with KNN-classification. I used also KNN with PCA to reduce the dimensionality and confusion matrix to see how good results were. With linear regression I got good prediction about humidity (mean square error= 1.136157472625289) but when I made dimensionality reduce result was much worse (mean square error=6.0247905559672885). With KNN I got quite good classification with accuracy 0.79. After reducing dimensionality result was same so it didn’t make result any better.

Introduction

When I started this project, I wanted to learn about PCA, linear regression and KNN-classification and how to use these in data processing. I also wanted to learn how to use these with python and how to make plots from results and understand the results. I have some interest in machine learning so topics in this project were perfect for me.

In project I was answering for questions about can we predict the relative humidity based on the available measurements with linear regression and from plots made from data. Also, I was answering question that could dimensional reduce make prediction better.

Other question I was answering was that can we classify conditions to be either as “dry" or “not dry", based on the available measurements. In this question I was looking answer with using KNN-classification. Also, I had to solve how many neighbours I would have to choose to get optimal result from KNN-classification. I also test if I could make classification better with dimensional reduce.

This task is important because its very useful to solve if we could predict or classify things from a large amount of data. Also, by doing this task I will learn about KNN-classification and linear regression which I find interesting.

Data Analysis:

Data is split in train and test and divided in four csv files:

• Weather_data_train: contains 3140 observations, and 16 columns.

• Weather_data_train labels: contains the labels ‘U mu’ and ‘OBSERVED’ for these 3140 observations.

• weather_data_test: contains 1346 observations, and 16 columns

• weather_data_test_labels: contains the labels “U mu" and “OBSERVED" for these 1346 observations

Data’s index is the datetime and numerical attributers are:

– “T" is the air temperature, in degrees Celsius, 2 meters above the earth’s surface.

– “Po" is the atmospheric pressure at weather station level, in millimetres of mercury.

– “P" is the atmospheric pressure reduced to mean sea level, in millimetres of mercury.

– “Ff" is the mean wind speed at a height of 10-12 meters above the earth’s surface, in meters per second.
– “Tn" is the minimum air temperature, in degrees Celsius, over the past day.

– “Tx" is the maximum air temperature, in degrees Celsius, over the past day. – “W" is the horizontal visibility, in km.

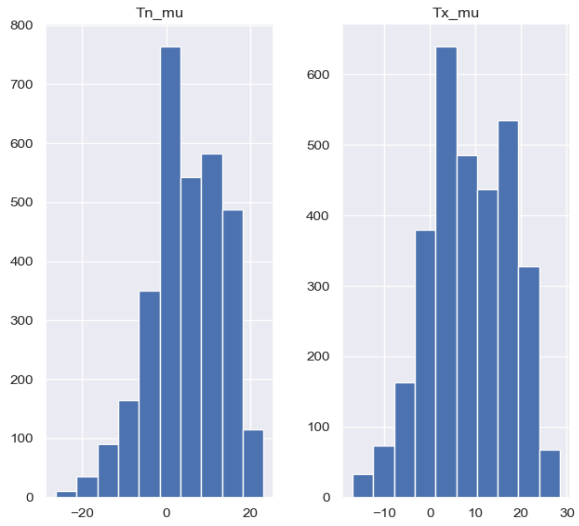
– “Td" is the dewpoint temperature at a height of 2 meters above the earth’s surface, in degrees Celsius.

– “U" is the relative humidity, in percentage, 2 meters above the earth’s surface.

– “OBSERVED" is a categorical variable, where 0 (not dry) indicates that the amount of precipitation was more than 0.3 millimetres, and 1 (dry) indicates that there was little or no precipitation.

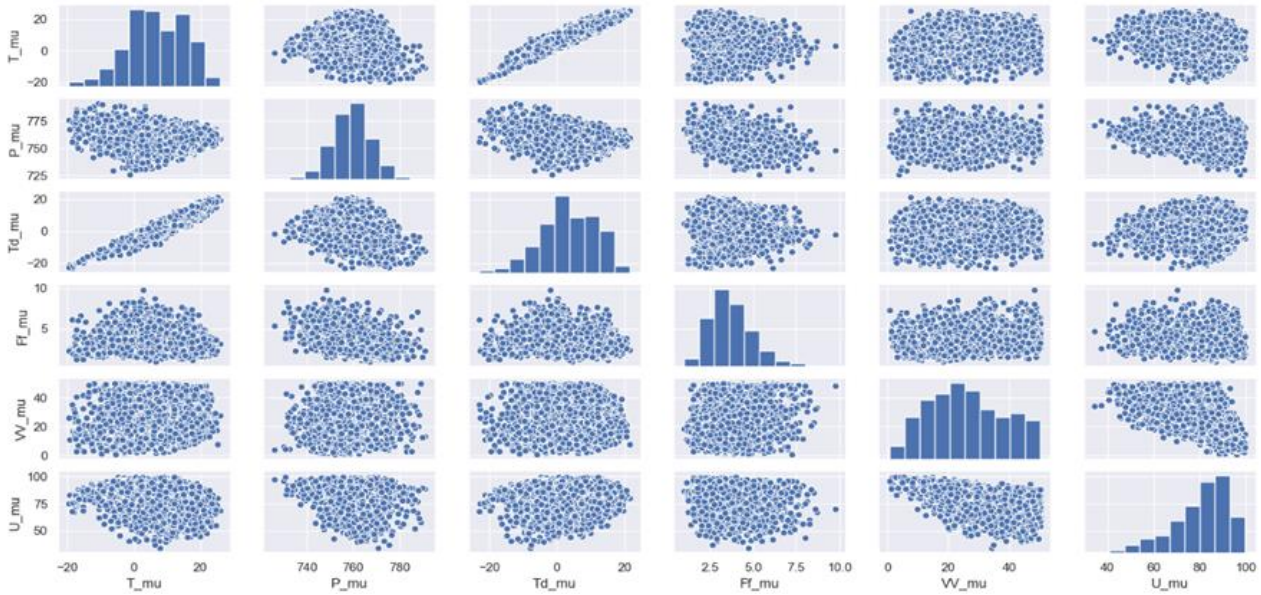
In data columns are for example Td_mn which mean value and Td_var which means variance.

Histogram of Tn_mu and Tx_mu:



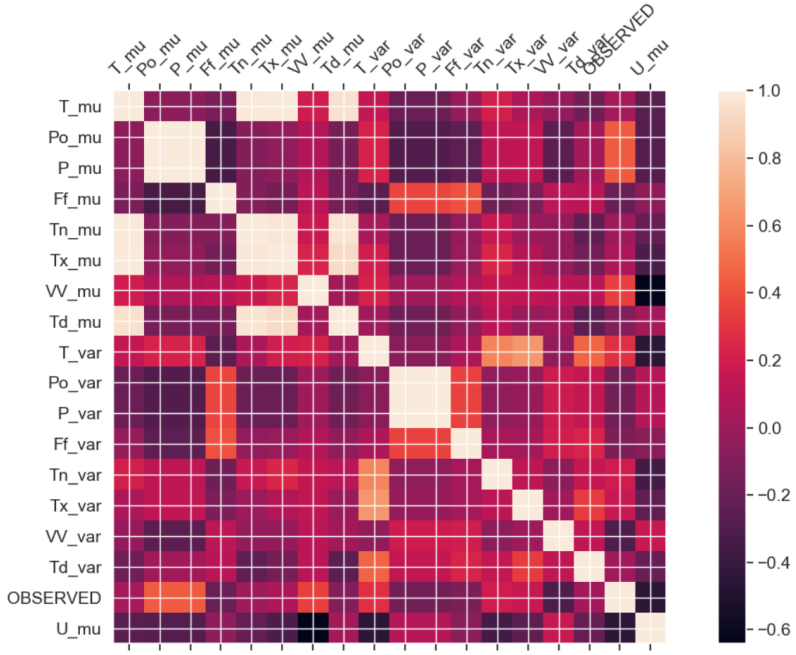
In histogram Y-axis means number of days and X axis means temperature in Celsius. We can see from histograms that they look quite same. This cloud tell that there have been only little temperature changes during days.

Pair plots for T_mu, P_mu, Td_mu, Ff_mu, VV_mu, U_mu:



We can see that there is clear relationship between Td_mu and T_mu because dots have settled linearly. There is also some connection between W_mu and U_mu because their dots have settled also mainly linearly. There is also little connection between P_mu and U_mu. I don't see any significant connection in other plots.

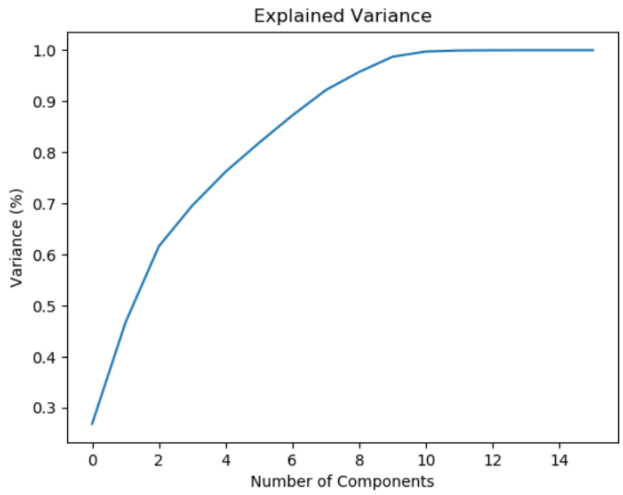
Correlation matrix of all features:



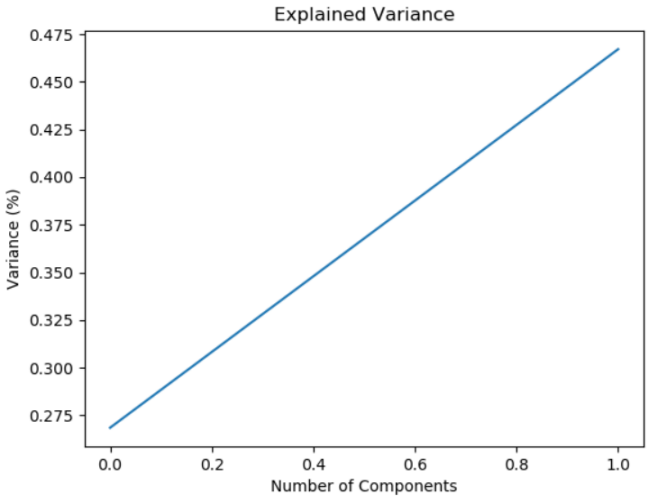
Can be seen from correlation matrix that there are big correlations between columns that are related with temperatures. There is also big correlation between U_mu and W_mu and also between U_mu and Td_mu.

PCA:

PCA plot of all components:



PCA plot of two components:



We can see from PCA plot of all components that with 9 components we can preserve about 97%-99% of all variance of the data so that is the optimal number of components that we should use in experiments.

Methods:

1. Histogram of Tn_mu and Tx_mu:

I used hist() function from matplotlib library to make histogram from Tn_mu and tx_mu.

2. Pair plot for T_mu, P_mu, Td_mu, Ff_mu, VV_mu, U_mu:

I used pairplot() function from seaborn library to make pair plot from T_mu, P_mu, Td_mu, Ff_mu, VV_mu and U_mu.

3.Correalation matrix of all features:

I connected weather_data_train and weather_data_train_labels to one data frame with concat() function from pandas library. After that I made correlation matrix of all features with corr() function from pandas library. Correlation matrix is a matrix that contains correlation coefficients between variables.

4.PCA

I made PCA projection with PCA function from library sklearn. I used the same function many times in experiments. Always before doing PCA I standardized the data with StandarScaler() function from library sklearn to get better results.

This is how PCA is done mathematically (Kumar, 2018):

1. Calculate covariance matrix of data points.

2. Calculate eigen vectors and corresponding eigen values

3. Sort eigen vectors according to their eigen values decreasing order

4. Choose first k eigen vectors and that will be new k dimensions.

5. Transform original n dim dimensional data points into k dimensions.

After doing PCA with PCA() function I made plots from explained variance of all components and from two components.

5. Regression

I used correlation matrix to see which features have most common with U_mu which means the relative humidity, in percentage, 2 meters above the earth's surface. This way I could find some predictors for humidity. I also used pair plots to see if there would be some similarities with U_mu.

I tried linear model to predict the relative humidity. I used linear_model function from sklearn library. I used weather_data_train and weather_data_train_labels to train linear model. After training I used weather_data_test to predict humidity and weather_data_test_labels to see how good prediction was. I used linear_model function because it was easy and convenient way to do linear regression and I got results fast.

This is how linear regression is done mathematically. When implementing linear regression of some dependent variable y on the set of independent variables $\mathbf{x} = (x_1, \ldots, x_r)$. There can be assumed that there is linear relationship between y and x $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon$. This equation is the regression equation and $\beta_0, \beta_1, \ldots, \beta_r$ are the regression coefficients**.** Linear regression calculates the estimators of the regression coefficients or simply predicted weights, denoted with $b_0, b_1 \ldots, b_r$. They define the estimated regression function $f(\mathbf{x}) = b_0 + b_1 x_1 + \cdots + b_r x_r$. This function should capture the dependencies between the inputs and output sufficiently well. With multiple independent variables, the case with my experiment, is similar as with two variables, but more general. The estimated regression function is $f(x_1, \ldots, x_r) = b_0 + b_1 x_1 + \cdots + b_r x_r$, and there are $r + 1$ weights to be determined when the number of inputs is r.(Stojiljkovic, 2019)

After making linear regression I used PCA function to reduce dimension of the weather_data_train and weather_data_test to see if there would come better results.

6. Classification

I took Observed column values to the one vector from weather_data_train_labels data to do the KNN-classification. I used KNeighborsClassifier from sklearn.neighbors library to do KNN-classification for OBSERVED column. I set number of neighbours to 16 and trained classifier with weather_data_train and vector with values from OBSERVED column. After this I classified weather_data_test values with KNN-classifier and I checked how good classification was from observed column of weather_data_test_labels data. After doing KNN-classification I made confusion matrix of the results with confusion_matrix() function from sklearn library.

This is how KNN algorithm works:

First step in KNN is to transform data points into feature vectors, or their mathematical value. The algorithm then works by finding the distance between the mathematical values of these points. It then finds the probability of these points being similar to the test data and classifies it based on which points share the highest probabilities. (Schott, 2019) Confusion matrix of the results will show that how many of the results were wrong and how many rights.

I made KNN also with dimension reduce by using PCA to see if I would get better result with it. I decided number of neighbours to be 39 in this case.

Experiments and Results:

Regression:

1. Strongest predictors for humidity:

We can see from correlation matrix and pair plot that U_mu which is mean relative humidity, in percentage, 2 meters above the earth's surface has some connection (correlation = -6.4) with W_mu which is mean horizontal visibility, in km. This is clearly one predictor for humidity. Also, t_var have relatively big correlation with U_mu (correlation=-0.44).

2. Linear model to predict humidity:

I made linear model with Weather_data_train, Weather_data_train labels, weather_data_test and weather_data_test_labels. I used train data to train model and test data to see if we can predict humidity with linear model. Below you can see table of sample from results. Observed column is rounded in table. As we can see predicting with linear model worked well and results were closely same as in original data. Variance score of y_test and y_pred was 0.65 which tells that pred and test have mainly same values but there is some kind of variation occurs. Mean square error of this prediction was 1.136157472625289. According this prediction is pretty good because value of the mean square error is small.

y_test		y_pred	
OBSERVED	U_mn	OBSERVED	U_mn
0	93.875	0	93.735
1	88.500	1	89.320
0	85.250	0	86.280
0	91.375	0	91.109
0	94.500	0	93.279
1	78.875	1	79.444
1	79.125	1	79.600
1	89.250	1	89.280
1	84.875	1	85.248
1	69.375	1	71.014

3. Linear model with PCA to predict humidity:

I made linear model with adopting PCA to x_train and x_test data to see if there would come better prediction for humidity. I chose the number of the components to be 9 according the PCA plot of all components which I introduced earlier. Results wasn't good as in linear model without PCA. Variance score was 0.42. Compared to variance score which I got without PCA (0.65) result of this predict is not so good. Below you can see sample of this prediction. Observed column is rounded to the nearest even number. We can see that result differs more than earlier linear model from original data. Mean squared error with this prediction was 6.0247905559672885 which is lot bigger than prediction done without PCA. This tells that this prediction isn't so reliable.

y_test		PCA_y_pred	
OBSERVED	U_mn	OBSERVED	U_mn
0	93.875	0	86.991
1	88.500	1	82.045
0	85.250	0	86.373
0	91.375	0	86.623
0	94.500	0	81.390
1	78.875	1	73.912

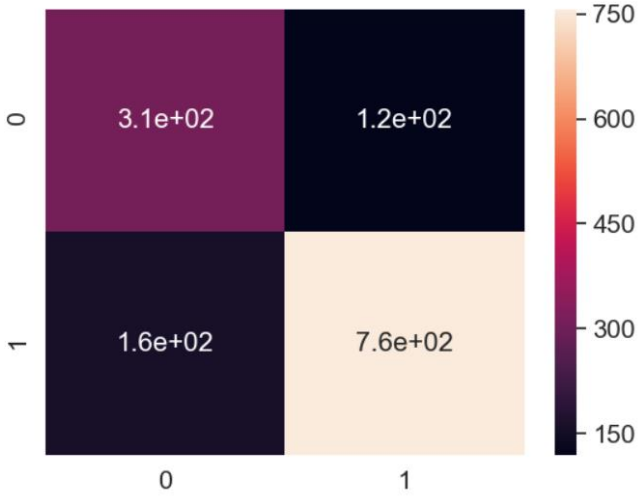
1	79.125	1	71.558
1	89.250	1	74.338
1	84.875	1	75.167
1	69.375	1	75.422

Classification:

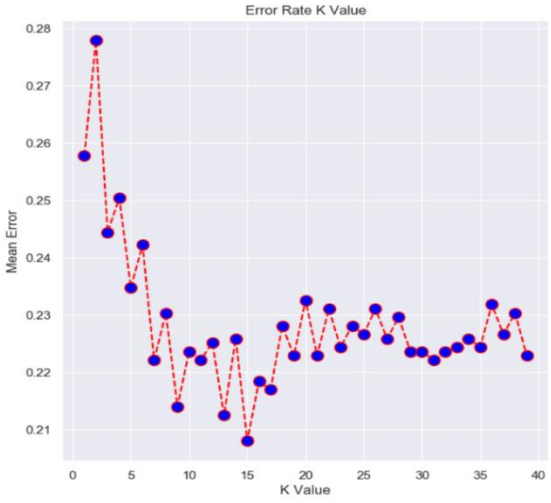
1. KNN-classification:

I made KNN-classification to classify the weather conditions either as “dry" or “not dry". I used wether_data_train and wether_data_train_labels to train KNN-classifier. Before training I removed U_mu column from wether_data_train_labels data because that was useless for this experiment. After training I classify conditions of weather_data_test with KNN-classifier and after that I compared it with original data from weather_data_test_labels. I made confusion matrix from result as you can see below. I used 15 neighbours in my KNN-classification. I decided optimal number of neighbours by counting with how may neighbours error rate is lowest. Below is a plot about error rates with k neighbours.

Confusion matrix:



Error rates:

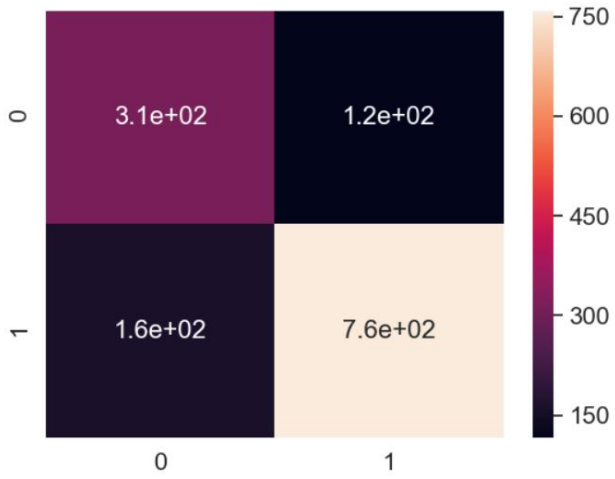


We can see from confusion matrix that accuracy of the classifier is (310+760)/1350 =0.79 and misclassification rate is (160+120)/1350=0.21 So with KNN classifier we can get pretty good classification about if the weather conditions are either “dry" or “not dry" but it will make some errors.

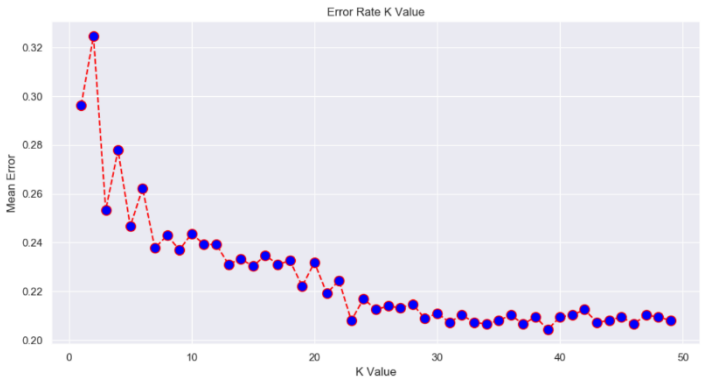
2. KNN- classification with PCA:

I chose the number of the components to be 9 according the PCA plot of all components which I introduced earlier. I made PCA for wether_data_train to reduce the dimension of the data set. I trained KNN-classifier with wether_data_train and wether_data_train_labels. After this I made PCA for with wether_data_test and put it in to the classifier. I compared results to the original data from weather_data_test_labels and I made confusion matrix about results which you can see below. I used 39 neighbours in my KNN-classification with PCA. I decided optimal number of neighbours by counting with how may neighbours error rate is lowest. Below is a plot about error rates with k neighbours.

Confusion matrix:



Error rates:



We can see from confusion matrix that accuracy of the classifier is (310+760)/1350 =0.79 and misclassification rate is (160+120)/1350=0.21 So with KNN classifier we can get pretty good classification about if the weather conditions are either “dry" or “not dry" but it will make some errors. Results were same as without dimension reduce with PCA.

Conclusion and discussion

Regression:

I made conclusion that strongest predictor for humidity is W_mu which means mean the horizontal visibility, in km. I made this conclusion because W_mu have strongest correlation with U_mu. This could be because when there is humidity there usually is also fog which will affect to the horizontal visibility.

Linear model to predict humidity:

From linear model I can make conclusion that we can predict humidity pretty well with it. Mean square error was only 1.136157472625289 which means that there is no huge error with prediction, however there could be some. But I would say that linear model gives pretty good prediction of humidity.

Linear model with PCA to predict humidity:

Linear regression with reducing dimension of the data set with PCA gives worse result than linear model without PCA. Mean square error was 6.0247905559672885 which is much bigger than without PCA. This gives us untrusted prediction of the humidity. I thought that I was having wrong number of components because of the poor result but when I tried with other amount of components result was no better. I can make conclusion that linear model gives much better prediction of the humidity without dimension reduce with this data. I think that dimension reduce did not give better result because there was no enough data. Also is possible that I made a mistake during experiment because linear regression should give better result with dimension reduce.

Classification:

We can make conclusion from KNN-classification that with KNN we can classify conditions to be either as “dry" or “not dry" quite well but there will be some error. I would say anyway that accuracy of the classifier (0.79) is good. It could be better if we would give classifier more data to classify conditions.

Reducing dimensionality with PCA didn't give better result. Result were same as without PCA. Maybe there was not enough data to make dimensional reduce improve results. Also, in this case is possible that I made a mistake during experiment because KNN-classification should give better result with dimension reduce.

References:

Kumar, Rishav. (2018). Understanding Principal Component Analysis, Medium[Online], Available at: https://medium.com/@aptrishu/understanding-principle-component-analysis-e32be0253ef0

Stojiljkovic, Mirko. (2019). Linear Regression in Python. Real Python[Online], Available at: https://realpython.com/linear-regression-in-python/

Schott, Madison. (2019). K-Nearest Neighbors (KNN) Algorithm for Machine Learning. Medium[Online], Available at: https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26

Gonsales, Arthur. (2018). An Approach to Choosing the Number of Components in a Principal Component Analysis, Towards Data Science[Online], available at: https://towardsdatascience.com/an-approach-to-choosing-the-number-of-components-in-a-principal-component-analysis-pca-3b9f3d6e73fe