Predicting PCOS with K-Nearest Neighbor

María de los Ángeles Contreras Anaya A01700284

***Abstract* -- This document presents the implementation of the K-Nearest Neighbor algorithm to predict Polycystic Ovary Syndrome and the performance of the algorithm against different features of the dataset and different values for K.**

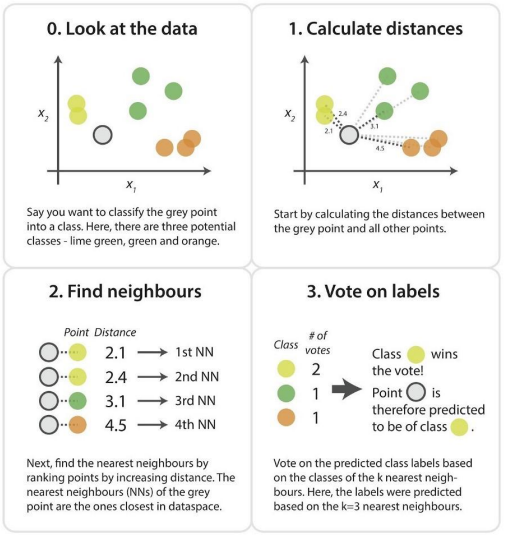
I. Introduction

Polycystic ovary syndrome (PCOS) is a condition that affects a woman's hormone levels, women with PCOS produce higher-than-normal amounts of male hormones, called androgens. PCOS impact between 8 and 20% of women in their reproductive age worldwide and half of the people that have it, are unaware of it. PCOS is one of the most misdiagnosed syndromes because the symptoms that come with it have a variety of potential causes, which makes it even harder to pinpoint the real cause. Technology, more exactly Artificial Intelligence, can be of great use in this field to be able to diagnose women with PCOS on time and be able to treat them in order to prevent other diseases like heart disease, diabetes, mental health conditions, reproductive disorder or cancer of the uterine lining.

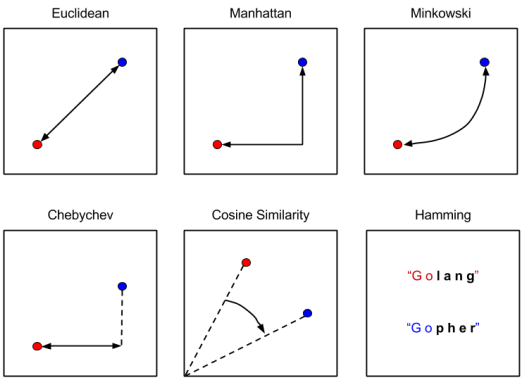
II. State of the Art

The method for pattern classification known as K-nearest neighbor was introduced in an unpublished US Air Force School of Aviation Medicine report in 1951by Fix and Hodges. This algorithm falls under the supervised learning category, and it can be used either for regression or classification. KNN is known as a lazy learning algorithm because it does not perform any training when data is supplied, rather the algorithm learns until a query is performed. KNN is also considered a non-parametric method because it does not make any assumptions about the distribution of the supplied data. The algorithm works by finding the distances between a query and all the labels in the data, then selecting the number (k) of labels closest to the query and finally

predicting or classifying the queried data, to do so the algorithm may perform a voting mechanism for the most frequent label (for classification) or calculating the average of the labels (for regression).



KNN uses different types of metrics for finding the distance between training values and the queried one, the one used by default by sklearn library is the euclidean distance which calculates the straight line distance between two points in a Euclidean space, but the algorithm may also use: the Manhattan, Minkowski, Chebychev or Hamming distance.



III. Dataset

The dataset used in this implementation was found on Kaggle, it contains data collected from ten different hospitals across Kerala, India. The set consists of 45 columns and 541 entries of type float and int that document the symptoms and some other metrics of the patients and whether they ended up having PCOS or not. The dataset was downloaded as an Excel file and had some missing or corrupted values that were calculated manually (ex. BMI calculated from weight and height values). The set was then exported to .csv and read using pandas library on python.

```
RangeIndex: 541 entries, 0 to 540
Data columns (total 45 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Sl. No                  541 non-null    int64
 1   Patient File No.        541 non-null    int64
 2   PCOS (Y/N)              541 non-null    int64
 3    Age (yrs)              541 non-null    int64
 4   Weight (Kg)             541 non-null    float64
 5   Height(Cm)              541 non-null    float64
 6   BMI                     541 non-null    float64
 7   Blood Group             541 non-null    int64
 8   Pulse rate(bpm)         541 non-null    int64
 9   RR (breaths/min)        541 non-null    int64
 10  Hb(g/dl)                541 non-null    float64
 11  Cycle(R/I)              541 non-null    int64
 12  Cycle length(days)      541 non-null    int64
 13  Marraige Status (Yrs)   540 non-null    float64
 14  Pregnant(Y/N)           541 non-null    int64
 15  No. of aborptions       541 non-null    int64
 16    I   beta-HCG(mIU/mL)  541 non-null    float64
 17  II    beta-HCG(mIU/mL)  541 non-null    object
 18  FSH(mIU/mL)             541 non-null    float64
 19  LH(mIU/mL)              541 non-null    float64
 20  FSH/LH                  541 non-null    float64
```

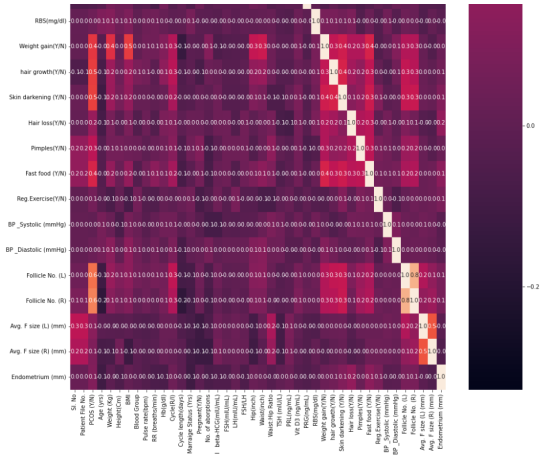
IV. Implementation

- **Feature selection**

  Feature selection is one of the most important steps before feeding data into machine learning algorithms, the right selection of features can increase the predictive power of the algorithm by removing both redundant data (noise) and misleading data, consequently it also reduces the training time. The feature selection technique used on this prediction was the correlation coefficient.

  Correlation is a measure that documents the linear relationship of two variables, the higher the correlation value, the easier it is to predict one variable from the other.

  In this case I used seaborn library which is built on top of matplotlib to display the correlation between all the features of the dataset.



For feature selection I am taking as the best features for my model, the ones with the highest correlation value to my target attribute which is PCOS. From the above plot, it is easy to see that the best features for my model are:

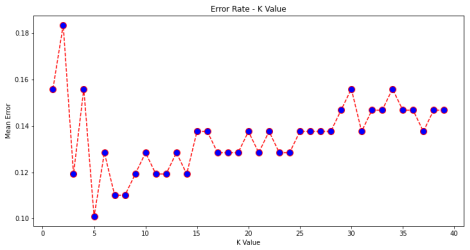
- Follicle No. (R)
- Follicle No. (L)
- Skin darkening
- Hair growth

Notice that the correlation between L and R is really high, on the testing phase of this paper we will be using just one of these features instead of both to see how it impacts the predictive power.

- **Finding the best "K"**

The K in KNN algorithm refers to the number of nearest neighbors to a particular data point (queried data) that will be taken into account in the classification or regression process. For the algorithm to work with a great accuracy, there is a need to tune the K parameter, namely choosing the right value for K.

To find the optimal value for K an error plot was used, the error plot displays the relationship between the k value and the error rate associated to it. The best value for the KNN algorithm in this case is the one with the minimum error rate.



From the plot, you can see that the smallest error we got is 0.59 at K=5.

- ## Classifying

Once the best parameters ('Follicle No. (R)', 'hair growth(Y/N)' and 'Skin darkening (Y/N)') and the optimal value for k (5) was found the dataset was split into training (80%) and testing(20%) data and then using the KNeighborsClassifier from sklearn the model was fitted and predictions were made.

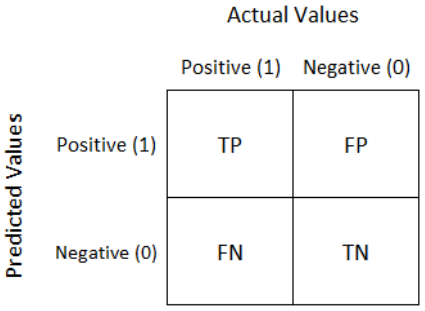
```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

# Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

# Applying K-NN algorithm
classifier= KNeighborsClassifier(n_neighbors=7)
classifier.fit(x_train, y_train)
y_pred= classifier.predict(x_test)
```

- ## Evaluating predictions

To evaluate the predictions of our model, a confusion matrix, classification report and the accuracy were used.

Accuracy is the ratio of correct predictions to total predictions made, but the metric does not give away any details on the performance of each class, which the confusion matrix does. Our model had an 89.91% accuracy.

A confusion matrix is a common technique used to summarize the performance of a classification algorithm. It is a better-suited technique for getting insight into how and why is the classification model getting "confused". The confusion matrix displays four different combinations of predicted and actual values.



Where:

- TP - True positive.
- TN - True negative.
- FP - False positive.
- FN - False negative.

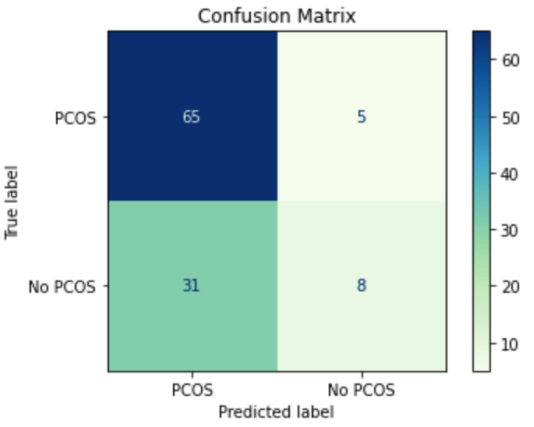
To calculate the confusion matrix we used the method confusion_matrix from sklearn and plotted the outcome, from the testing portion of our dataset which is a compound of 109 entries, the model predicted 98 of them correctly.

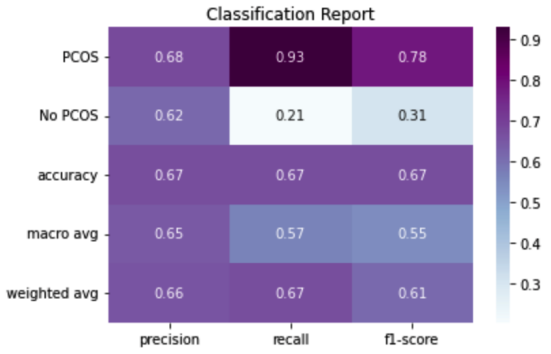
Furthermore, from our classification report we can visualize the precision, recall and f1 score of our model, in this case, it is crucial to detect the syndrome, so we want to have the tiniest amount of false negatives, that is why one of the best metric to evaluate our model is recall, which is the measure of our model correctly identifying True Positives. From the classification report we can see that our models recall was even better that our models' accuracy, recall was of 94%.

V. Tests

The first test was done using four of the worst features that were included in the dataset, which were: 'FSH/LH', 'Cycle length(days)', 'PRG(ng/mL)' and 'RBS(mg/dl)' with an arbitrary value for K = 15.

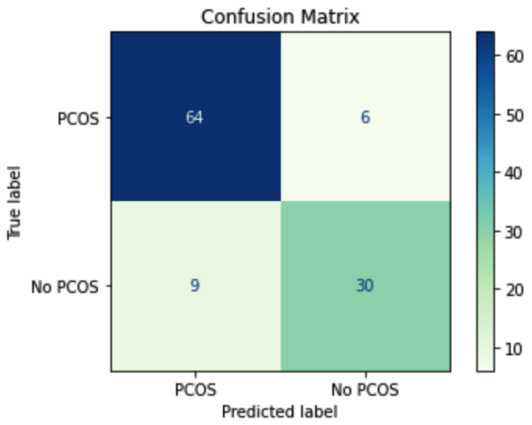
The performance of this combination of features + k value was not great, since the accuracy of the predictions was of 66.97%. Furthermore, from the predictions made, only 73 were correct and the rest were falsely predicted.

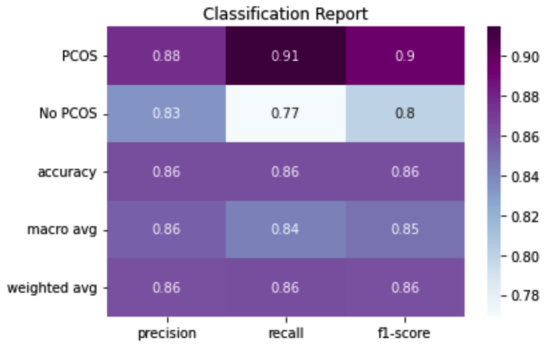




The second test was done using the best four features included in the dataset, which were: 'Follicle No. (R)', 'Follicle No. (L)', 'hair growth(Y/N)' and 'Skin darkening (Y/N)' with the same arbitrary value for k used in the first test 15.

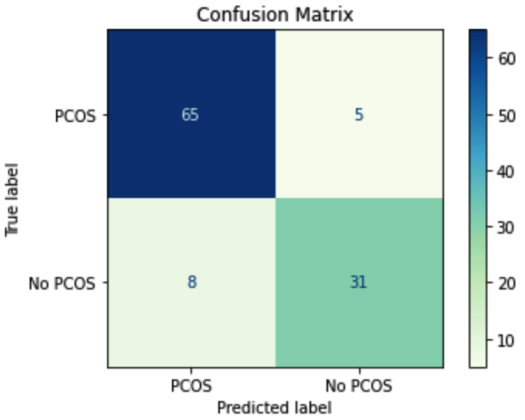
The performance of this combination of features + k value improved the model significantly since the accuracy of the predictions was of 86.24% which is far better from the first results. Furthermore, from the predictions made, 94 out of 109 were correct. This test proves that feature selection is key to improving prediction power.

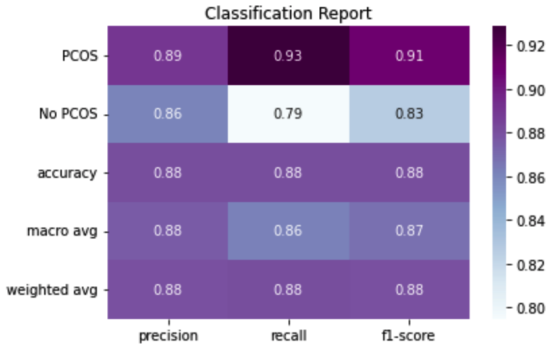


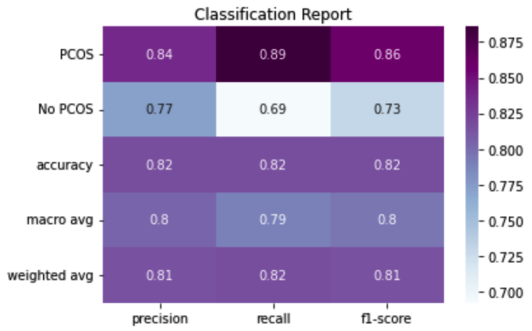


The third test was done using the best four features except for those that were highly correlated to each other, which ended up being: 'Follicle No. (R)', 'hair growth(Y/N)' and 'Skin darkening (Y/N)') with the same value for K = 15.

The performance of this combination of features + k value was the best one yet, since the accuracy of the predictions was of 88.07%.

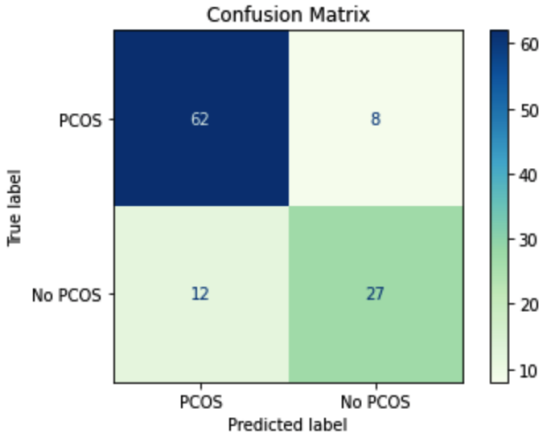




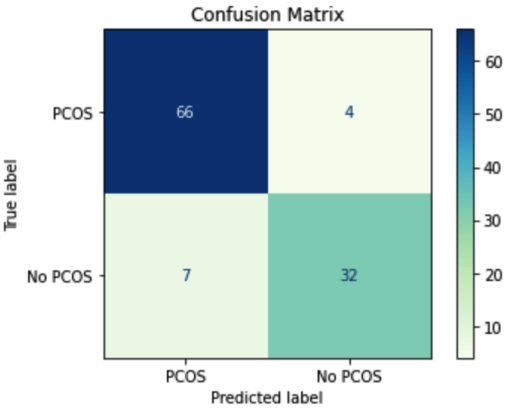


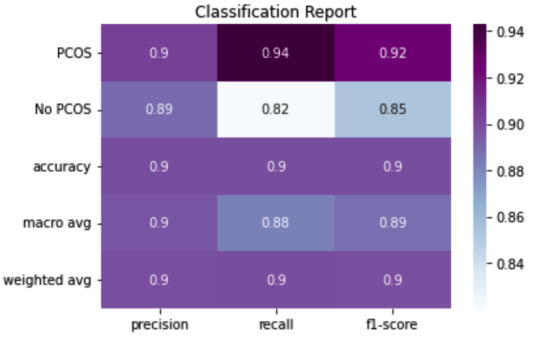
The fourth test was done using the features that performed better in the lasts tests, which were: 'Follicle No. (R)', 'hair growth(Y/N)', 'Skin darkening (Y/N)' and the worst value for K that we found during implementation phase (2).

The performance of this combination of features + k value decreased, the accuracy of the predictions was of 81.65%, 6 points below above test. Selecting an optimal value for K is just as important as feature selection.



Lastly, the final test was done using the best features and the best value for k according to our error plot, which was 5. This test presented an accuracy of 89.91% the maximum value of all tested scenarios and greatest f1-score (92 from a .9 precision and a .94 recall measure) which represents an amazing improvement in the prediction power of the model.





VI. Conclusions

From the test that were conducted during the implementation of the KNN algorithm to predict patients with PCOS we can summarize our learnings:

1. Feature selection is, from the results we got, the most important step when designing a machine learning model in order for the algorithm to be truly effective. In our first test (worst features) we had a pretty great recall (94%) for diagnosing PCOS, but our precision was far from great so probably our algorithm just diagnosed almost every entry for a patient as positive, having a better set of feature we improved that precision to 88 and our accuracy from 67% to 86%.
2. Having correlated features can in fact be detrimental to the performance of our algorithm, but it is not as bad as having a bad set of features. This can be observed by comparing the second and third test, in the second test we had two highly correlated (0.8) features (Follicle No. (R) and Follicle No. (L)) an accuracy of 86% and in the third test we removed one of the previous features and ended up having an accuracy of 88% and an average of two points improvement on both cases for recall and precision metrics.
3. The selection of an optimal value for K is just as important as choosing a right set of features, this was proved by our fourth test where we used the best performance set of features so far, but we changed the value for k to the worst one (2) according to our error plot that evaluated error rate from 1 to 40. In this test we got an accuracy of 82% approximately 6 points below the accuracy of the third test and the worst performance in diagnosing correctly patients with no PCOS.

VII. References

Band, A. (2020, May 23). *How to find the optimal value of K in KNN? - Towards Data Science*. Medium; Towards Data Science. https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb#:~:text=The%20optimal%20K%20value%20usually,be%20aware%20of%20the%20outliers.

Feature Selection Techniques in Machine Learning. (2020, October 10). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

Laurenti, G. (2020, November 25). *Confusion Matrix and Classification Report - The Startup - Medium*. Medium; The Startup. https://medium.com/swlh/confusion-matrix-and-classification-report-88105288d48f

Peterson, L. (2009). K-nearest neighbor. *Scholarpedia*, *4*(2), 1883. https://doi.org/10.4249/scholarpedia.1883

Prasoon, K. (2020). *Polycystic ovary syndrome (PCOS)*. Kaggle.com. https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos

Precision vs Recall | Precision and Recall Machine Learning. (2020, September 3). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

Raj, A. (2021, April 8). *Introduction to Classification Using K Nearest Neighbours*. Medium; Towards Data Science. https://towardsdatascience.com/getting-acquainted-with-k-nearest-neighbors-ba0a9ecf354f

Sarang Narkhede. (2018, May 9). *Understanding Confusion Matrix - Towards Data Science*. Medium; Towards Data Science. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62