Patient Diagnosis with Machine Learning

Caitlin Ruble

July 2022

# Introduction

# Problem:

For a sick patient, the pathway to feeling better begins with an accurate diagnosis. Doctors are the first line responders to assess a patient’s symptoms, categorize the patient with a diagnosis, and then create a treatment plan based on that diagnosis. While this approach can, and often does work, and has been the approach for the entirety of the history of medicine, it does require doctors to hold vast amounts of memorized knowledge, doesn’t account for emerging diseases, and is at the mercy of human error. When ailments are misdiagnosed, patients suffer and cannot get the treatment they need. At best, they suffer for longer than they need to, and at worst their ailments can lead to permanent disabilities and even death. What if we could leverage all the canonical knowledge of the field of medicine and ask a machine to hold on to that for us, instead of relying on individual doctors to memorize and apply their knowledge perfectly? If we could build a machine learning model that correctly classified a patients diagnosis based on their presenting symptoms, we could save lives and reduce suffering by ensuring timely, accurate diagnoses, thus leading to timely, appropriate treatment. This is the problem we sought to solve!

Client:

Such a classifier would add monetized value to many entities across the medical industry. Patients who receive an accurate diagnosis and appropriate treatment the first time around will avoid unnecessary medical charges related to treating a misdiagnosis and are likelier to stay out of emergency situations related to their ailment. This equates to significant cost savings for hospital systems, health insurance providers, and other medical provider agencies, all while delivering better outcomes for patients.

Dataset:

The "Disease Prediction Using Machine Learning" dataset on Kaggle contains symptom data for 4962 individual observations. There are 132 binary symptom features, and a “prognosis” feature which holds the disease classification label. It should be noted that the dataset was found to be of poor quality during the course of the project, but I elected to complete the project as a proof of concept anyway, with the caveat that real patient data of sufficient quantity must be used to validate the ML models before they are ready to get anywhere close to predicting diagnoses for real patients.

Approach:

I will we

# Data Handling

Description of Features:

The following is a complete list and description of the features included in the dataset:

*Files: Training.csv and Testing.csv*

**prognosis**: the categorical assignment of a disease label for each observation, our target. The 41 diagnoses included in the data set are listed in Table 1, below.

**Unnamed: 133:** an extra column containing all NaN values

**symptom features:** 132 binary feature columns, each containing a 0 to indicate absence and a 1 to ­indicate presence of the symptom for each observation. The individual feature names are listed in Table 2, below.

|  |  |  |
| --- | --- | --- |
| **Table 1:** List of the 41 disease labels included in the original data set under feature name “prognosis” | | |
| AIDS  Acne  Alcoholic hepatitis  Allergy  Arthritis  Bronchial Asthma  Cervical spondylosis  Chicken pox  Chronic cholestasis  Common Cold  Dengue  Diabetes | Drug Reaction  Fungal infection  GERD  Gastroenteritis Heart attack hepatitis A Hepatitis B  Hepatitis C  Hepatitis D  Hepatitis E  Hypertension  Hyperthyroidism | Hypoglycemia  Hypothyroidism  Osteoarthristis  Paralysis (brain hemorrhage)  Peptic ulcer diseae  Pneumonia  Psoriasis  Tuberculosis  Typhoid  Urinary tract infection  Varicose veins  (vertigo) Paroymsal Positional Vertigo |

Data Cleaning:

The .csv files were loaded into a Jupyter notebook as a pandas DataFrame. The test set provided in the testing.csv file only contained 1-2 instances for each class, so the data were concatenated and shuffled with the Training.csv data, leading to a total of 4962 entries. All columns were checked for the correct data type, and no changes to the data types were necessary.

Handling of Missing Values:

1. There was an extra feature column, ‘Unnamed: 133,’ which was found to contain all NaN values. This column was dropped from the data set.
2. The symptom feature ‘fluid\_overload’ was found to contain only 0s for the entire data set, indicating it was adding no information. Additionally, there was a redundant symptom feature named ‘fluid\_overload.1’ which *did* contain binary information. The original ‘fluid\_overload’ column was deleted and the ‘fluid\_overload.1’ column was renamed to ‘fluid\_overload’ for ease of interpretation. This brought our symptom feature count to 131.

Handling of Formatting and Misspelling Issues:

Based on domain knowledge and careful review of the prognosis categories, several inconsistencies that could present interpretation issues in a production environment were identified and remedied. These formatting changes were carried out on the ‘prognosis’ feature cells, and are summarized in Table 3.

Handling of Duplicate Observations:

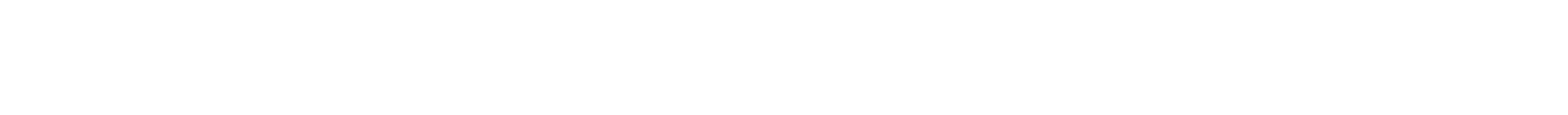
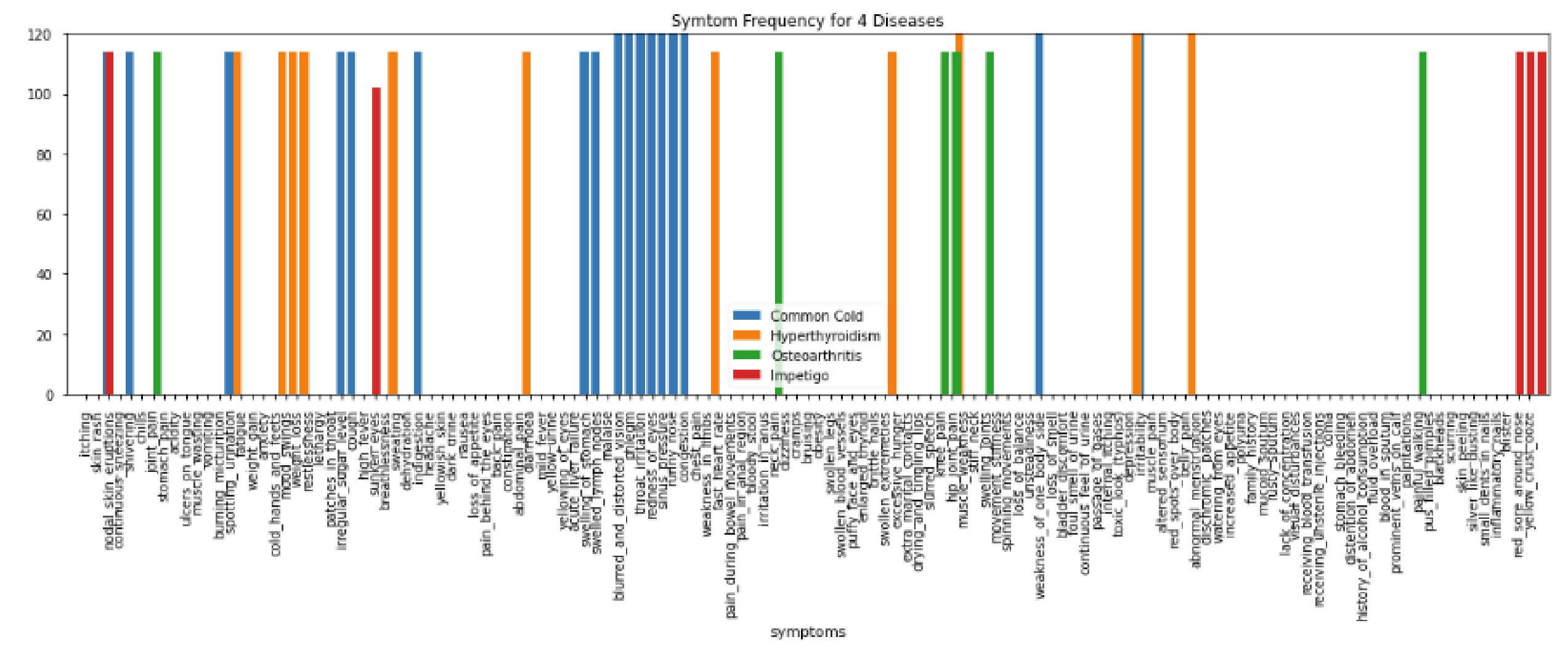
The data was analyzed for duplicate row values, and was found to contain *mostly* duplicate values. The maximum value of original observations for a prognosis was 10, and the minimum was 5. The decision was made to continue with the modeling operations for the sake of demonstration, however the data is virtually useless. The models will, however, be ready when data of higher quality and volume is found.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 2:** List of binary symptom feature columns in the original data set | | | | | |
| abdominal\_pain abnormal\_menstruation  acidity  acute\_liver\_failure altered\_sensorium anxiety back\_pain belly\_pain blackheads bladder\_discomfort blister  blood\_in\_sputum bloody\_stool  blurred\_and\_distorted\_vision breathlessness brittle\_nails bruising  burning\_micturition chest\_pain  chills  cold\_hands\_and\_feets coma congestion constipation  continuous\_feel\_of\_urine continuous\_sneezing cough cramps dark\_urine dehydration depression diarrhoea dischromic \_patches  distention\_of\_abdomen  dizziness  drying\_and\_tingling\_lips enlarged\_thyroid excessive\_hunger extra\_marital\_contacts family\_history fast\_heart\_rate fatigue fluid\_overload fluid\_overload.1  foul\_smell\_of urine headache | | high\_fever hip\_joint\_pain  history\_of\_alcohol\_consumption increased\_appetite indigestion inflammatory\_nails internal\_itching  irregular\_sugar\_level irritability irritation\_in\_anus  itching joint\_pain knee\_pain  lack\_of\_concentration lethargy loss\_of\_appetite loss\_of\_balance loss\_of\_smell malaise mild\_fever mood\_swings movement\_stiffness mucoid\_sputum muscle\_pain muscle\_wasting muscle\_weakness nausea neck\_pain  nodal\_skin\_eruptions obesity  pain\_behind\_the\_eyes pain\_during\_bowel\_movements pain\_in\_anal\_region painful\_walking palpitations passage\_of\_gases patches\_in\_throat phlegm polyuria  prominent\_veins\_on\_calf puffy\_face\_and\_eyes pus\_filled\_pimples  receiving\_blood\_transfusion receiving\_unsterile\_injections red\_sore\_around\_nose  red\_spots\_over\_body redness\_of\_eyes | | restlessness runny\_nose rusty\_sputum scurring shivering silver\_like\_dusting sinus\_pressure skin\_peeling skin\_rash slurred\_speech small\_dents\_in\_nails spinning\_movements spotting\_ urination stiff\_neck stomach\_bleeding stomach\_pain sunken\_eyes sweating  swelled\_lymph\_nodes  swelling\_joints swelling\_of\_stomach swollen\_blood\_vessels swollen\_extremeties swollen\_legs throat\_irritation toxic\_look\_(typhos) ulcers\_on\_tongue unsteadiness visual\_disturbances vomiting watering\_from\_eyes weakness\_in\_limbs weakness\_of\_one\_body\_side weight\_gain weight\_loss yellow\_crust\_ooze yellow\_urine yellowing\_of\_eyes yellowish\_skin | |
| **Table 3:** Formatting changes to disease classifications in ‘prognosis’ feature | | | |
| **Original** | | **Cleaned** | |
| ‘hepatitis A’ | | ‘Hepatitis A’ | |
| ‘Osteoarthristis’ | | ‘Osteoarthritis’ | |
| ‘Dimorphic hemmorhoids(piles)’ | | ‘Dimorphic hemorrhoids (piles)’ | |
| ‘(vertigo) Paroymsal Positional Vertigo’ | | ‘Paroxysmal positional vertigo’ | |
| ‘Peptic ulcer diseae’ | | ‘Peptic ulcer disease’ | |

# Exploratory Data Analysis

Distributions of Symptoms for Each Disease:

Frequency plots of each symptom assigned to each disease were plotted, revealing that each disease category has a distinct combination of symptoms associated with it, creating a visually distinct “signature.” We know that each disease can have up to 120 counts of each symptom. When a frequency bar reached 120 (the top of the plot), it indicated that every single instance of that disease in the training data was positive for that symptom. While this isn't exactly rare (check out the "Common cold" or "Hyperthyroidism" charts, among others, for examples), it was far more common in our data set for many, but not all, of the instances of a disease to show a particular symptom. No symptom frequency bar was less than ~100 instances out of 120. Each prognosis had multiple symptoms associated with it. Several demonstrative examples are reproduced in Figure 1, below.

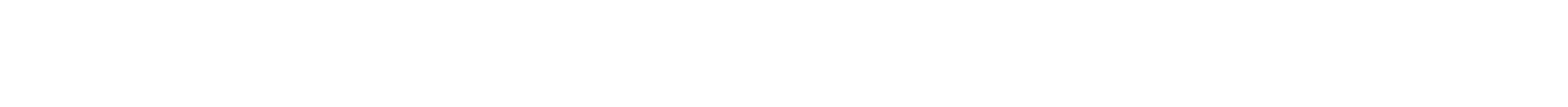
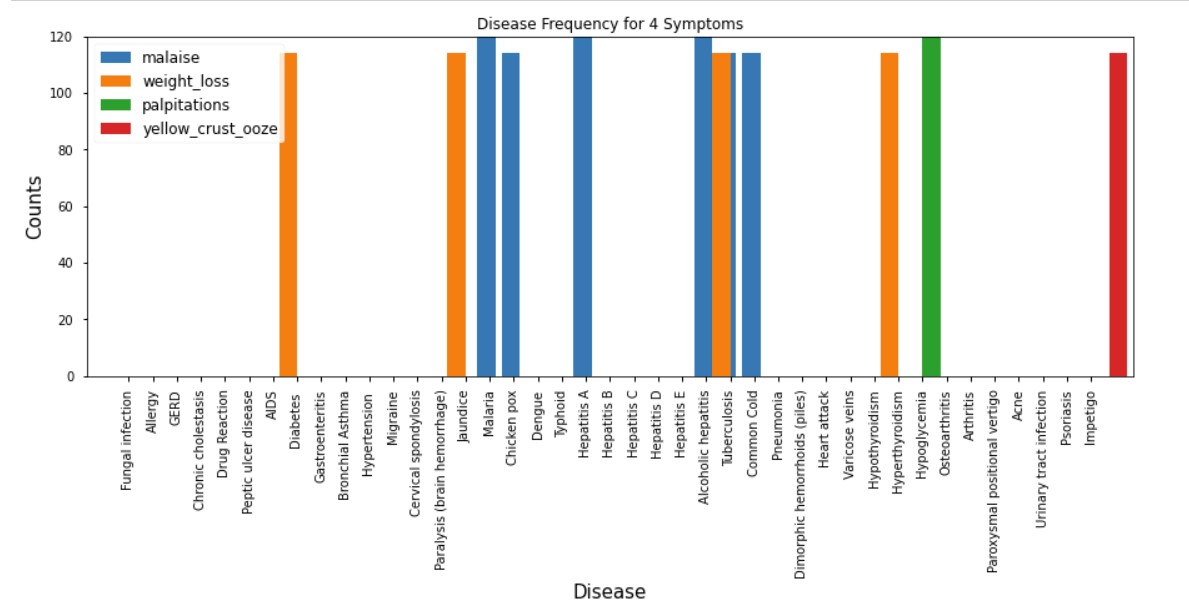


*Figure 1: Symptom Frequency Chart for 4 Diseases: Common Cold, Hyperthyroidism, Osteoarthritis, Impetigo. These are meant to be demonstrative of the different symptom “signatures” of each disease in the dataset showed, and how while some symptoms are implicated in 100% of the cases of a disease, many are not.*

Distributions of Diseases Associated with Each Symptom:

Frequency plots of which diseases were associated with each symptom were also plotted. Figure 2 shows a representative sample of 4 of the symptom features and which diseases were associated with them at which frequencies. There are several conclusions we drew:

1. Some symptoms were only indicated in 1 disease prognosis. (See ‘yellow\_crust\_ooze’)
2. Some symptoms were only indicated in 1 disease prognosis *and* are indicated in every instance of that prognosis in the training set. (See ‘palpitations’)
3. Some symptoms were indicated in more than one prognosis. (see ‘weight\_loss’)
4. Some symptoms were indicated in more than one prognosis and were always indicated in one or more of the possibilities. (see ‘malaise’)



*Figure*

*2*

*:*

*Fre*

*quency plot for the*

*occurrence*

*of*

*each disease for symptoms*

*'malaise',*

*'weight\_loss', 'palpitations', and*

*'yellow\_crust\_ooz*

*e'*

*. These 4 symptoms are representat*

*i*

*ve of the 4*

*categories*

*d*

*esc*

*ribed in the section above.*

Predictive Power Score of Each Symptom:

Calculating the Predictive Power Score of each feature on the target variable was conducted using the ppscore API available in Python from 8080 Labs1. This measure was an attractive option to quantify the correlation of each feature on the categorical target variable. A ppscore of 1 indicates that a feature column has perfect predictive power, while a score of 0 indicates a feature column has no predictive power. The maximum predictive power score was found to be 0.0041, and 20 symptoms scored here. 51 of the symptoms had a predictive power score of 0.00, indicating no predictive power. The remaining 60 symptoms scored between 0.0035 and 0.0021. This test showed us that no one feature was strongly correlated with the target feature (‘prognosis’), and, indeed, that no feature on its own has strong predictive power. If dimensionality reduction was desired, the 51 features with ppscore of 0 would be good candidates, however in the machine learning models we tested there was no need for this. These results are summarized in Table 4, below.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 4: Summary of Predictive Power Scores for Each Symptom Feature on Target Variable | | | |
| **Predictive Power Score** | 0.0022 | 0.0002 – 0.0016 | 0 |
| **Number of Symptom Features** | 20 | 60 | 51 |
| **Interpretation** | Best predictive power | Some predictive power | No predictive power |

1 8080 Labs PPS Repo can be found on Github at this address: https://github.com/8080labs/ppscore

# Data Preprocessing and Training Data Development

Data Encoding:

The symptom feature columns contained binary data, which were already formatted similarly to one-hot encoding. The target feature column contained categorical data, and was label encoded to map each distinct disease category to a distinct value for use with machine learning modeling.

Data Splitting:

The data were split into X and y, with X holding the 131 binary symptom features and y holding the label encoded categorical target feature. The data were further split into a training and testing set, with 80% of the data retained for training the machine learning models and 20% being retained to test each model’s performance.

# Modeling

Machine Learning Algorithms Implemented in SciKitLearn:

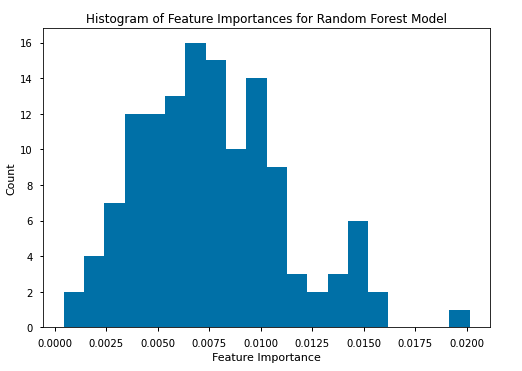
A total of 7 machine learning models were cross-validated with the training data: An entropy based decision tree, a gini-impurity based decision tree, a random forest classifier, an XGBoost classifier, a gradient boosted classifier, and Ada boosted classifier and a support vector classifier with an RBF kernel. In each case, an “off-the-shelf” version of the model was tried first before attempting hyperparameter tuning. The exceptions to this rule were the entropy-based and gini-impurity based decision tree models, which were tuned in their first implementation due to commonly known overfitting issues in unbound implementations. These models were compared on the basis of accuracy during 5-fold cross-validation with the training data. Ultimately, hyperparameter tuning was only performed on the decision tree classifiers and the Ada boost classifier, because every other classifier tested showed 100% (or near 100%) accuracy in cross-validation. The validation results are summarized in Table 5, below.

|  |  |
| --- | --- |
| Table 5: 5-Fold Cross-Validation Results for Machine Learning Model | |
| *Model* | *Mean CV Accuracy* |
| Random Forest | 1.0 |
| SVC | 1.0 |
| XGBoost | 1.0 |
| Gradient Boost | 1.0 |
| AdaBoost | 0.991 |
| Gini Tree | 0.943 |
| Entropy Tree | 0.928 |

Model Selection:

Based on cross-validation accuracy results, the four top performing models were the random forest, SVC, XGBoost, and Gradient Boost models, which each gave a perfect 100% accuracy in cross-validation. While this indicated great success in the classification problem, it gave rise to new questions. Namely, did we make a mistake and accidentally leave the target feature column in the independent variable set? Two pieces of evidence suggested this was not the case:

1. The three lowest-performing models were below 100% accuracy. If the high accuracy of the best performing models was an artifact of leaving the target variable in the independent variable feature space, we would expect all models to show 100% accuracy.
2. We investigated the feature importance values for one of the top performing models, the random forest classifier. The feature importance is a calculated value that explains how important a given feature is to the overall prediction; we expect a reasonable model to contain multiple feature importances, each a fraction of 1.0, the sum of which is 1.0. All feature importances of the random forest classifier were quite low and the sum across the feature space was 1.0, indicating that all features carried some weight in the classification and no one feature dominated the model’s prediction. A histogram of the feature importances can be seen in Figure 3, below.



*Figure 3: The histogram of feature importances shows that no one feature carried undue weight in the prediction of the classifier.*

At this point, we also checked the data for any duplicate values. We found that the data was almost entirely composed of duplicate values: the minimum unique row count per class was 5 and the maximum was 10. This meant that no class in our multi-class classification problem had more than 10 unique observations. Upon further review, we noticed that the data from Kaggle had no documentation referring to its source, and that other people working with the data had also achieving 100% accuracy. We concluded that the data was most likely a poorly-generated synthetic dataset with no modeling value whatsoever. Ideally this would have been caught in the data cleaning and exploration step, however this data scientist was early in her learning journey. A difficult lesson was learned—you ***must*** ensure your data quality! “Garbage in, garbage out.” Because most of the work for this project was already complete, we decided to proceed through to completion, with the large caveat that the data used must be replaced before any conclusions can really be drawn or decisions be made from the data. This poor data quality made up of duplicate observations explains the apparently high accuracy of the tested models.

We also did a final test on each model using the held-out test data and this time observed accuracy, F1 score, precision and recall, all of which came out to 1.0 (see Table 6, below). In order to make a selection on the “best model” out of the four top-performing models, we turned to another measure: model training time and sample prediction time. We measured both, with results summarized in Table 7, below.

If we deploy a model that continuously retrains every time new data comes in, then the training time metric would be the most important to optimize. Because this is a medical diagnostic tool, we would probably retrain in careful batches to maintain control over the model, so training time is less important. By this measure, the SVC Classifier is superior, taking just 0.49s to fit to our training data, and the Random Forest model is right behind at 1.01s.

We were more interested in how quickly a model could predict a diagnosis for a given occurrence. This metric would indicate how quickly a medical profession could expect to retrieve a diagnosis classification after inputting a patient’s symptom data in the model in a production environment. By this measure, the XGBoost classifier shows superiority, with the lowest test time of just 0.02ms per patient.

The model chosen would depend on how the model is deployed: the XGBoost Classifier has the quickest prediction time, but 60x the training time compared to the SVC model. The training time might be prohibitively slow for retraining the model as the quantity of data increases, as it would in a production environment. This could potentially be offset by utilizing XGBoost's parallel processing feature in a production environment, however that would take additional resources, and given equally accurate choices, it is good practice to choose that which can be implemented most simply.

Sitting in between XGBoost and SVC is the Random Forest Classifier. With a training time of 1.01s and test time of 0.09ms per patient, this model blends quick turn-around with quick training and is therefore the recommended model to move into production.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 6: Test Statistics for Tested Machine Learning Model | | | | s |
|  | Accuracy | F1 | Precision | Recall |
| Random Forest | 1.0 | 1.0 | 1.0 | 1.0 |
| SVC | 1.0 | 1.0 | 1.0 | 1.0 |
| XGBoost | 1.0 | 1.0 | 1.0 | 1.0 |
| Gradient Boost | 1.0 | 1.0 | 1.0 | 1.0 |

|  |  |  |
| --- | --- | --- |
| Table 7: Model Training Time and Sample Prediction Time Results | | |
|  | Training Time (s) | Prediction Time per Patient (ms) |
| Random Forest | 1.01 | 0.09 |
| SVC | 0.49 | 0.11 |
| XGBoost | 29.32 | 0.02 |
| Gradient Boost | 55.64 | 0.10 |

# Conclusions and Next Steps

The off-the-shelf machine learning models in sklearn by and large did amazingly well in predicting patient diagnoses in our testing data. We found 4 machine learning models that performed diagnosis classification at 100% accuracy on the test set, and selected the Random Forest Classifier as the recommended model to use in a production setting.

The 100% accuracy score is too good to be true in actual industry settings; upon investigation it was determined that the data was most-likely poorly-generated synthetic data and definitely was composed of mostly repeated observations. Therefore, this modeling process should be taken as **proof of concept** that such a diagnostic tool **can** be built to make accurate diagnoses, but further revision and testing with real patient data is absolutely required.

To improve the strength of the model, we suggest the necessity of finding and using real patient data. As more data is gathered, the model can be retrained with the extensive real-world data to challenge the model and extend its functionality.

When pushing the model through to production, make sure to use LabelEncoder’s inverse\_transform function on the predicted values to return the interpretable disease names rather than label encoded values.

We could explore targeting diagnoses that have very similar symptoms and therefore are more difficult for a doctor to differentiate. We could also explore adding in more-nuanced features, such as continuously valued test results or imaging features.

Extending this type of modeling into a recommender system or using predict\_proba in sklearn to return a series of possible diagnoses with associated probabilities could help the diagnostic tool to be more readily accepted by practicing medical professionals. After all, diagnosis has been an art form, and diagnosticians are unlikely to happily give that up. However, a useful tool that could return several likely diagnoses in order of probability would still give the diagnostician some space for human interpretation. This would be good at least in the beginning of using such a tool, until widespread trust and proven effectiveness can be established with real patient data.