Machine learning project

Malek Kanaan

Submitted to dr.

April 2024

Table of Contents

[1. Collecting Data 3](#__RefHeading___Toc78_2538187222)

[2. Preparing the Data 3](#__RefHeading___Toc191_2538187222)

[3. Choosing a Model 3](#__RefHeading___Toc193_2538187222)

[4. Training the Model 5](#__RefHeading___Toc195_2538187222)

[5. Evaluating the Model 6](#__RefHeading___Toc145_3632077514)

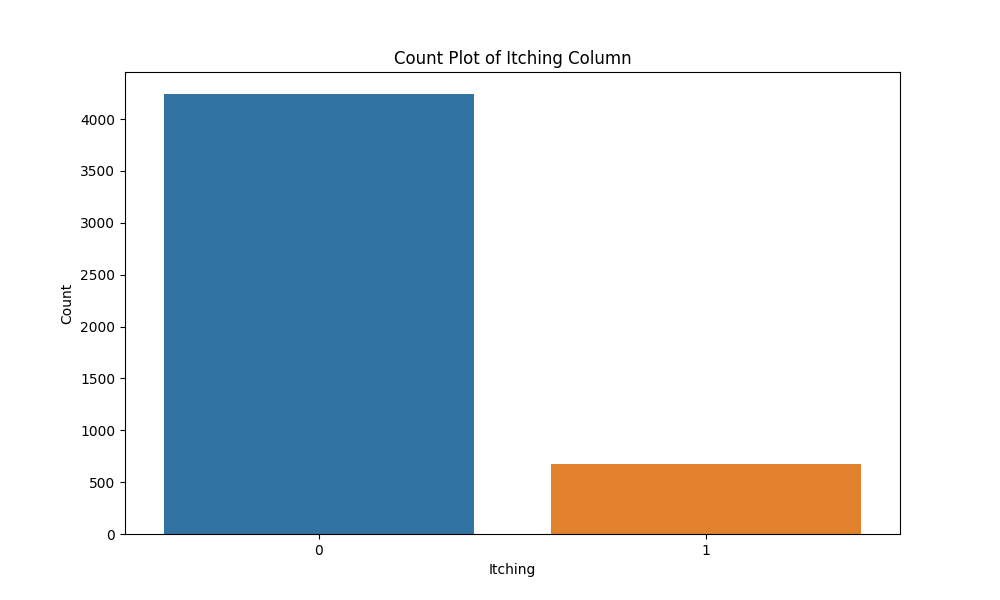
[6. Making Predictions 7](#__RefHeading___Toc201_2538187222)

# 1. Collecting Data

Downloaded, from kaggle, a onehot encoded and labeled dataset containing 132 common symptoms and 1 label column “prognosis”. The aim of this dataset is to create a classification AI model to predict a diagnosis (give a prognosis) to a suffering client. [Here is the link](https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning)!

# 2. Preparing the Data

Considering that the data is already onehot-encoded, all that was left was to label-encode the labels and to scale all the data by standardizing to increase training efficiency and accuracy. For sure, we cannot proceed with the training process if there is a problem with the data, such as existing empty rows or columns, splitting into validation and training sets, and randomizing the order of the entries. To visualize the data, we can draw countplots showing the frequency of the occurrences of each feature, here is an example to show the frequency of the itching feature:



# 3. Choosing a Model

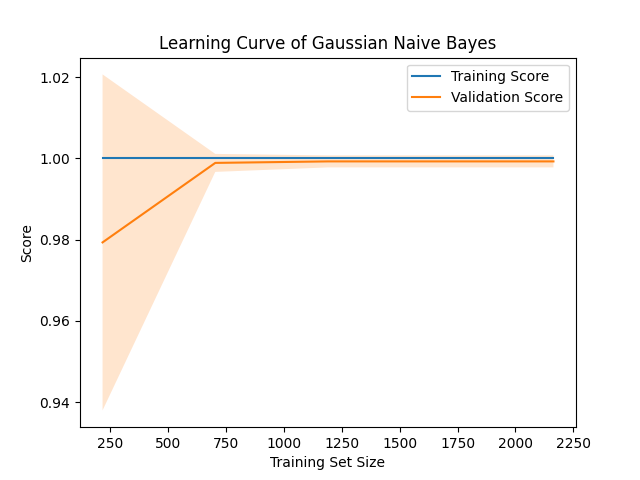
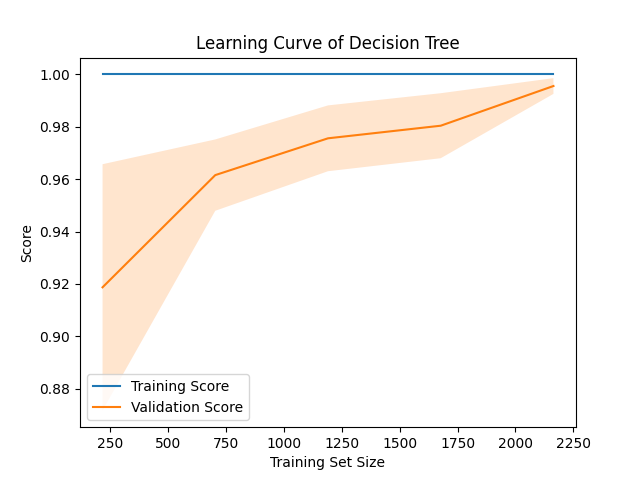
Choosing a model is not straight forward as it may seem.

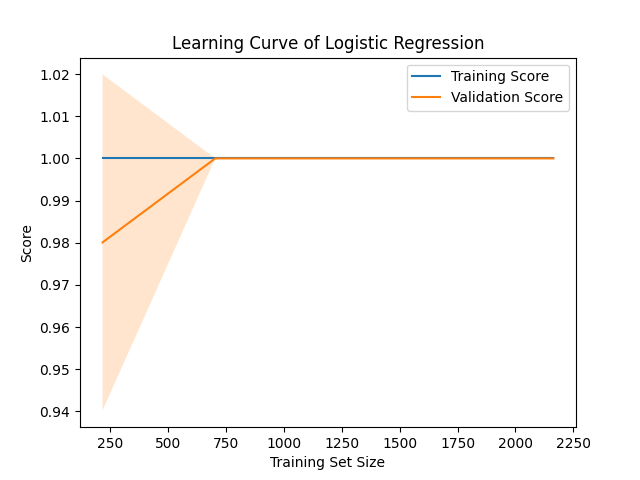
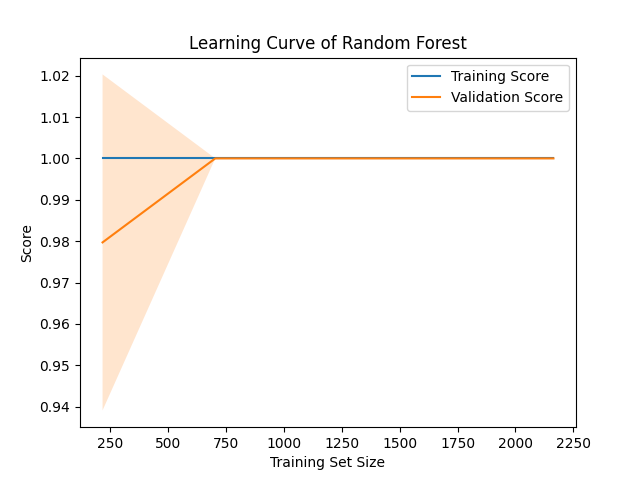
To begin with, we are working with a classification task using categorical features, this means that decision trees and random forests should excel. However, any machine learning classifier should work, like support vector machine, mutlinomial logistic regression, and naive bayes models.

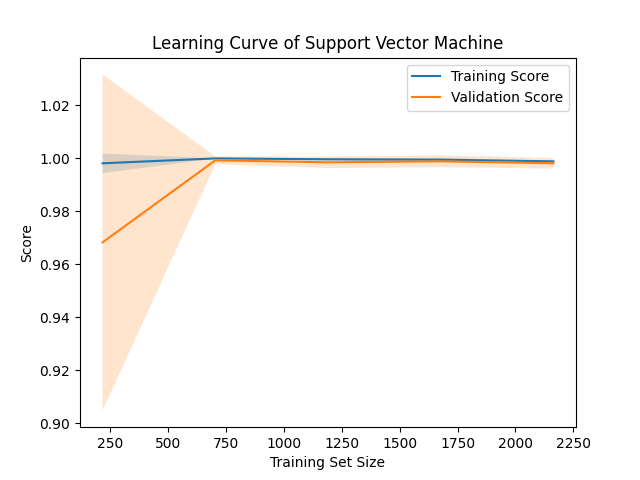
Therefore, we implemented these 5 models and employed the idea of ensemble learning, where we use these 5 models to make predictions in parallel, then what we can do is either pass the results into a softmax layer to decide which class to predict from the ensemble, as it would be the most occurring class, or by simply showing the user a percentage chance of each unique prediction, so we can have 1 100% prediction, 1 80% and 1 20%, all the way to 5 20% predictions. We chose to do the latter.

# 4. Training the Model

Scikit-Learn offers developer-friendly methods to train the models. A training loop is as simple as importing the type of model we want to train, including some optional parameters with the training data, then using model.fit(), then we can use joblib to save the trained model for future use. Here are the learning curves for each model:

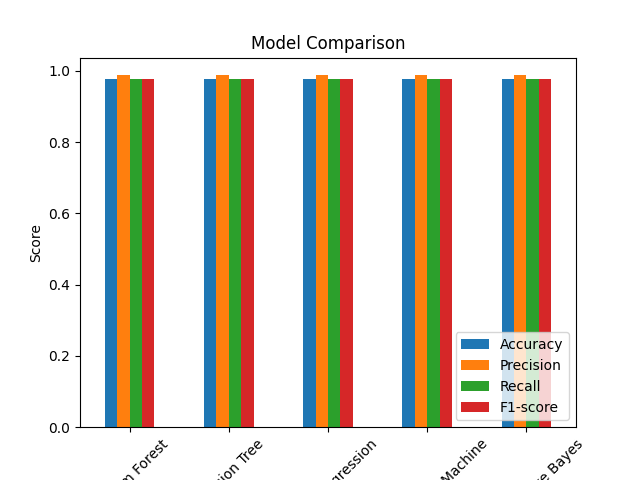




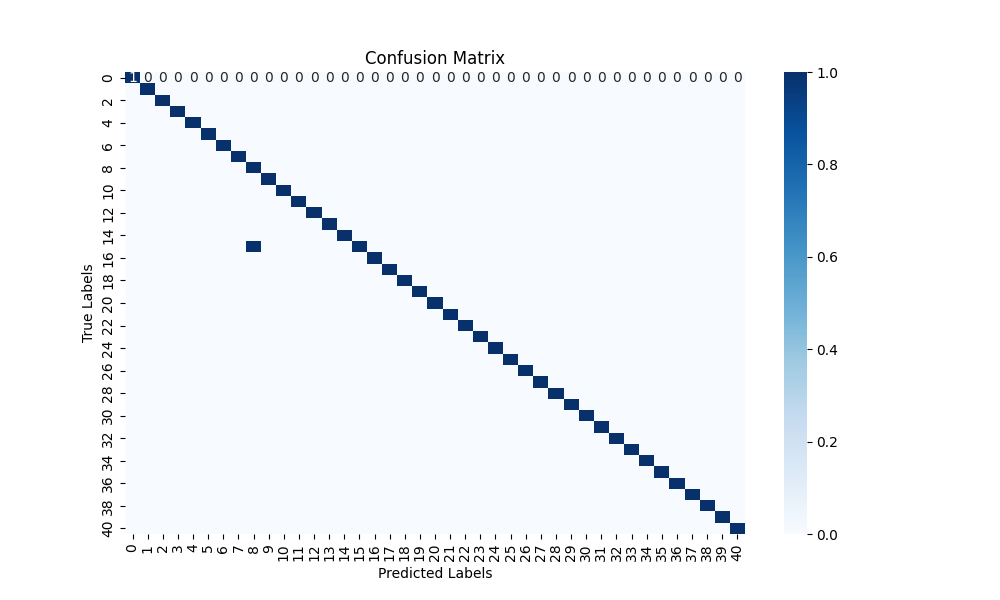


# 5. Evaluating the Model

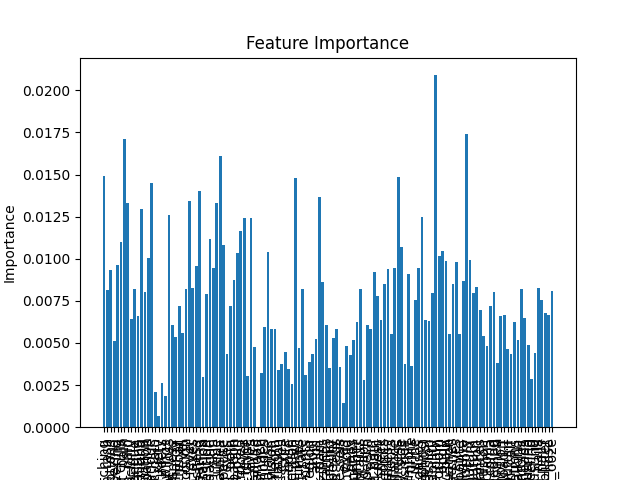
A model’s performance can be evaluated using a validation set, a testing set, as datapoints to be used to make dummy deployment predictions, using evaluation metrics like the f1-score, recall, precision, and accuracy.



Moreover, we can also see the confusion matrix, to verify that the testing set predictions are indeed accurate:

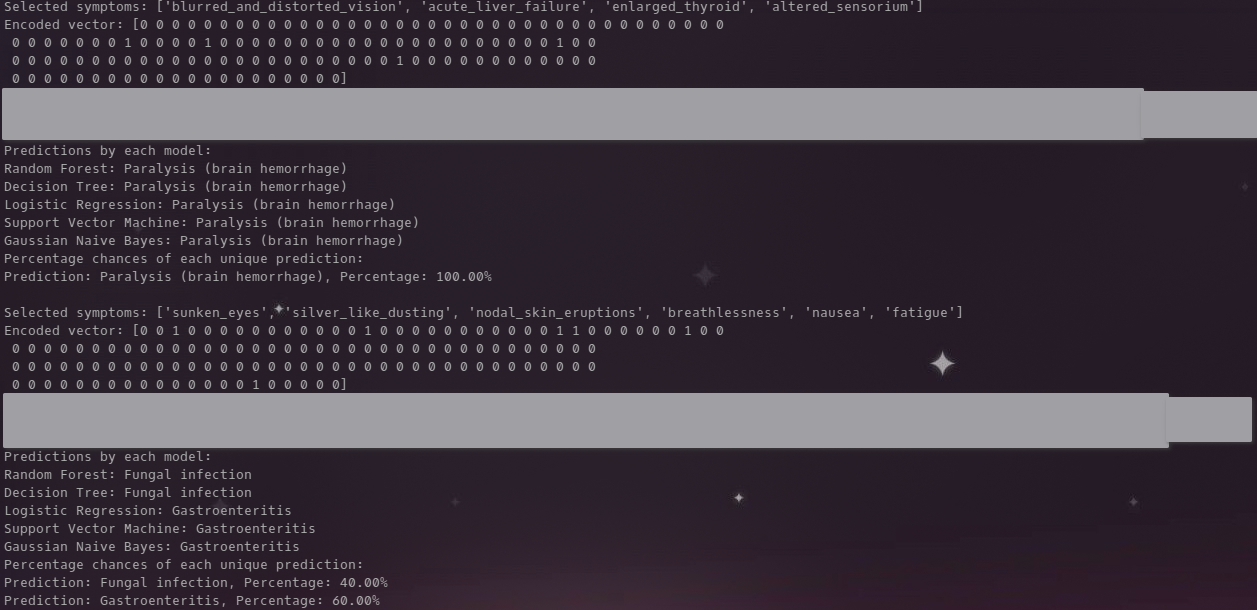


We can plot the feature importance according to the decision tree to get more insights about the data itself and how it influences its predictions, but that’s only optional:



# 6. Making Predictions

Implementing a random generator of symptoms, we can use the ensemble of models to make predictions on unseen data and to show the most possible predictions as discussed above:



Here is a text-copied output (different run so different output) since the image taken isn’t very clear:

**Selected symptoms:** ['lethargy', 'enlarged\_thyroid', 'yellow\_crust\_ooze', 'lack\_of\_concentration', 'pain\_behind\_the\_eyes', 'irritability']

**Encoded vector:** [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]

**Predictions by each model:**

Random Forest: Impetigo

Decision Tree: Hypertension

Logistic Regression: Hypertension

Support Vector Machine: Impetigo

Gaussian Naive Bayes: Hypertension

**Percentage chances of each unique prediction:**

Prediction: Impetigo, Percentage: 40.00%

Prediction: Hypertension , Percentage: 60.00%