Wellness Navigator- ”Disease Prediction using Ensemble Learning Techniques: A Comparative Study”

Siddhesh Dhavale

INDEX

PROBLEM

The problem that we are trying to solve for the Disease Prediction Dataset is to build a model that can accurately predict the disease prognosis based on a set of symptoms experienced by an individual. The dataset contains information on 42 different diseases and their corresponding symptoms. The goal is to train a machine learning model on the training dataset that can effectively learn the relationships between the symptoms and the corresponding disease prognosis. Once the model is trained, it can be used to predict the disease prognosis for new individuals based on their set of symptoms. This can help in early detection and diagnosis of diseases, which can lead to better treatment outcomes and improved patient care.

DATA SOURCE

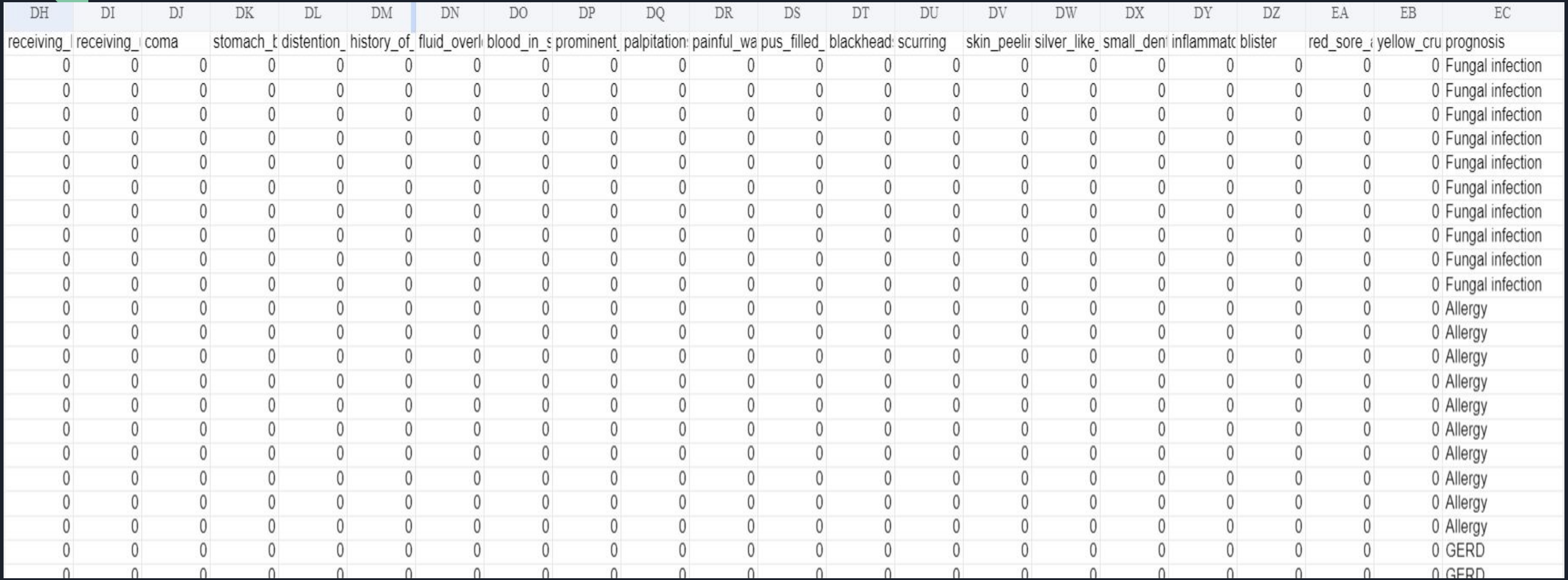
We have 2 data files - Training and Testing

Each file has 132 symptoms columns and a prognosis column as a result of symptom

Data could have been collected from medical records or surveys of individuals who have reported experiencing symptoms and have been diagnosed with a particular disease.

It is also possible that the data could have been artificially generated for research or testing purposes.

DATASET



DH	DI	DJ	DK	DL	DM	DN	DO	DP	DQ	DR	DS	DT	DU	DV	DW	DX	DY	DZ	EA	EB	EC
receiving_	receiving_	coma	stomach_b	distention_	history_of_	fluid_overl	blood_in_s	prominent_	palpitation	painful_wa	pus_filled_	blackhead	scurring	skin_peeli	silver_like_	small_den	inflammato	blister	red_sore_a	yellow_cru	prognosis
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GERD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GERD

DATA CLEANING

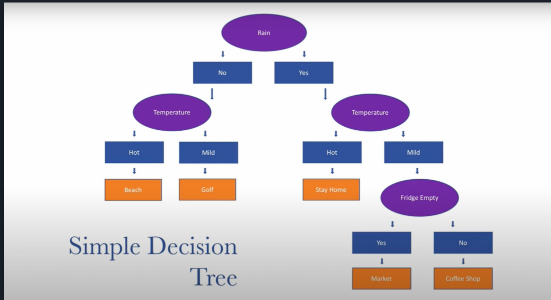
- Identify missing values: Check the dataset for missing values and determine if they can be imputed or if the corresponding observations need to be removed.
- Check for inconsistent values: Check for any inconsistencies in the data (e.g. where a binary variable has a value other than 0 or 1), dropped unnamed columns.
- Identify outliers: Check for any extreme values or outliers in the data. Outliers can be caused by errors in data entry, measurement error, or extreme values that are not representative of the general population
- Validate the data: After cleaning the data, it is important to validate that the data is consistent with the domain knowledge and the expectations for the variables.
- Normalize the data: The binary variables should be normalized to ensure that the values have a consistent meaning across the entire dataset.

SUPPORT VECTOR MACHINE

- SVM works by finding the best boundary (called a hyperplane) that separates the different classes of data in a high-dimensional space.
- SVM is useful for handling complex data with multiple features and can handle both linear and nonlinear classification tasks.
- The key idea behind SVM is to maximize the margin between the hyperplane and the closest data points, known as support vectors.
- SVM uses a kernel function to map the data into a higher-dimensional space where it can be linearly separable. Examples of kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.
- SVM has several advantages over other classification algorithms, including its ability to handle large datasets, high accuracy, and robustness against overfitting.
- SVM also has some limitations, such as its sensitivity to the choice of kernel function and parameters and its inability to handle imbalanced datasets well.

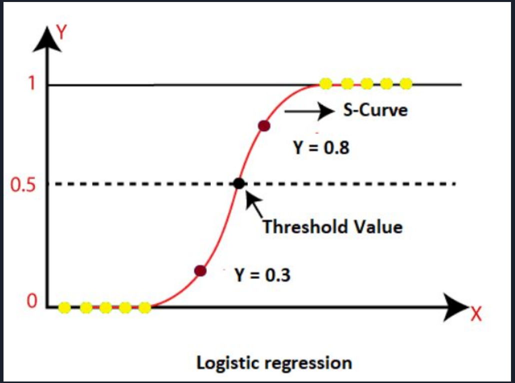
DECISION TREES/RANDOM FOREST

- Decision trees and random forests are supervised learning algorithms used for both classification and regression problems, random forests are a bunch of decision trees combined.
- The information gain metric is an information theoretic measure of how much “entropy” is revealed by a specific attribute. Tells us how important is a given attribute of the feature vector.
- Gini index is measured by subtracting the sum of squared probabilities of each class from one, whereas information gain is obtained by multiplying the probability of the class by log (base= 2) of that class probability.
- ID3 - It recursively selects and splits an attribute based on information gain measure; attribute that provides the maximum IG is placed at the root of the tree



LOGISTIC REGRESSION

- It's a method for predicting a categorical dependent variable from a set of independent variables
- Logistic regression models the relationship between the predictor variables and the binary outcome using a logistic function, which maps the predictor variables to a probability value between 0 and 1.
- The logistic function is S-shaped, meaning that it starts out steeply at the low end (when the predictor variables are low) and then levels off as it approaches the high end (when the predictor variables are high).
- The logistic regression model estimates the parameters of the logistic function using a technique called maximum likelihood estimation, which involves finding the parameter values that maximize the likelihood of the observed data given the model.
- Once the logistic regression model has been trained on a set of training data, it can be used to make predictions on new data by inputting the predictor variable values and obtaining a predicted probability of the binary outcome.
- The predicted probability can be thresholded at a certain level (e.g., 0.5) to make a binary classification decision. Not good for continuous outcomes.

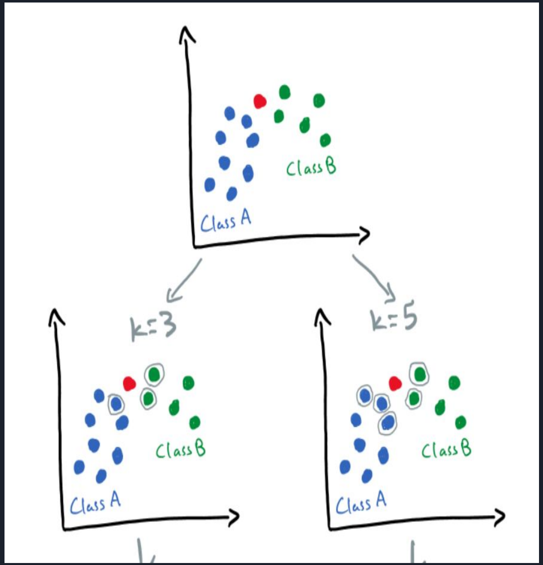


NAIVE BAYES

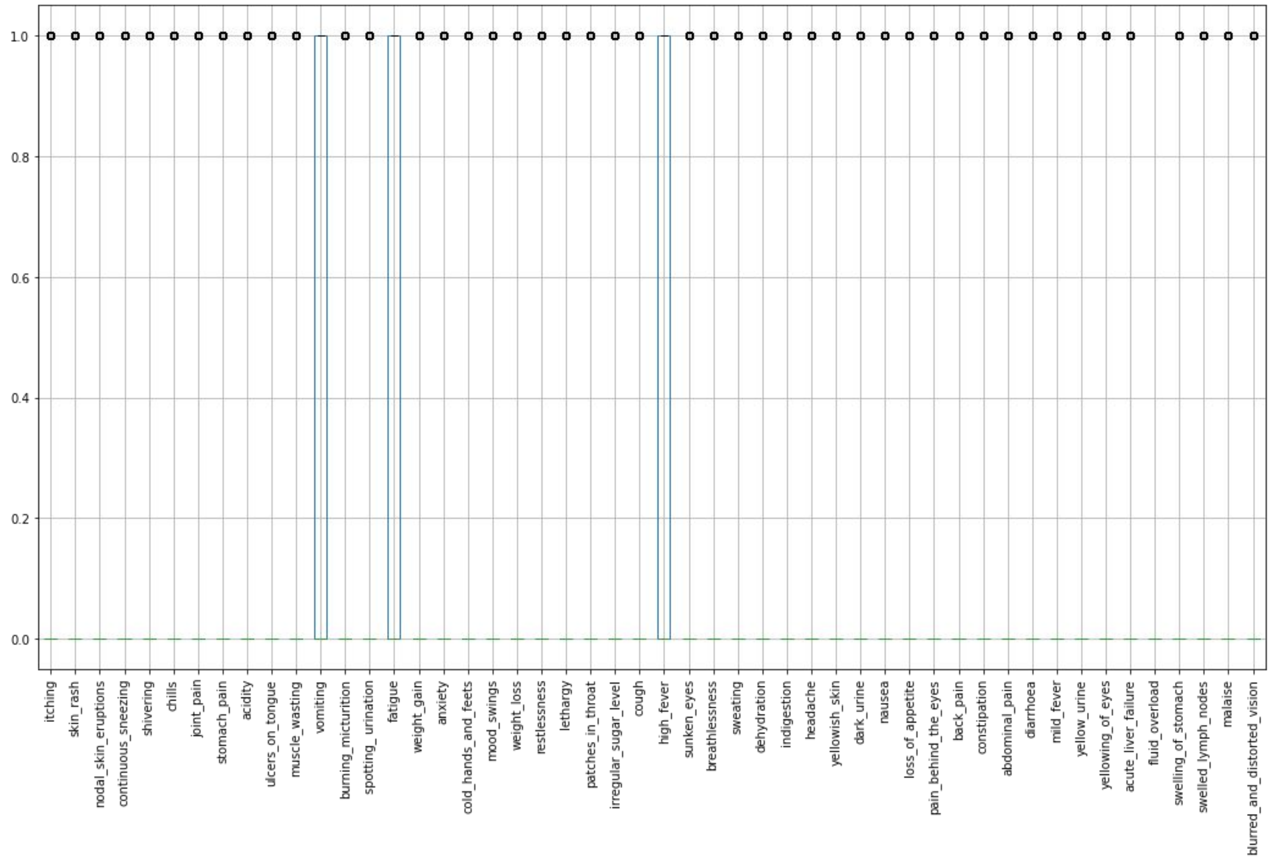
- It is based on Bayes' theorem, which is a way of calculating the probability of a hypothesis based on prior knowledge.
- Naive- it assumes that the occurrence of a certain feature is independent of the occurrence of other features
- The probabilistic algorithm works by first calculating the prior probability of each class, and then using Bayes' theorem to calculate the posterior probability of each class given the observed features.
- To classify a new instance, Naive Bayes simply chooses the class with the highest posterior probability.
- Naive Bayes is fast and efficient, making it a popular choice for large-scale classification tasks. However, it may not perform as well as more complex algorithms like support vector machines or neural networks on certain types of datasets.

K-NEAREST NEIGHBOUR

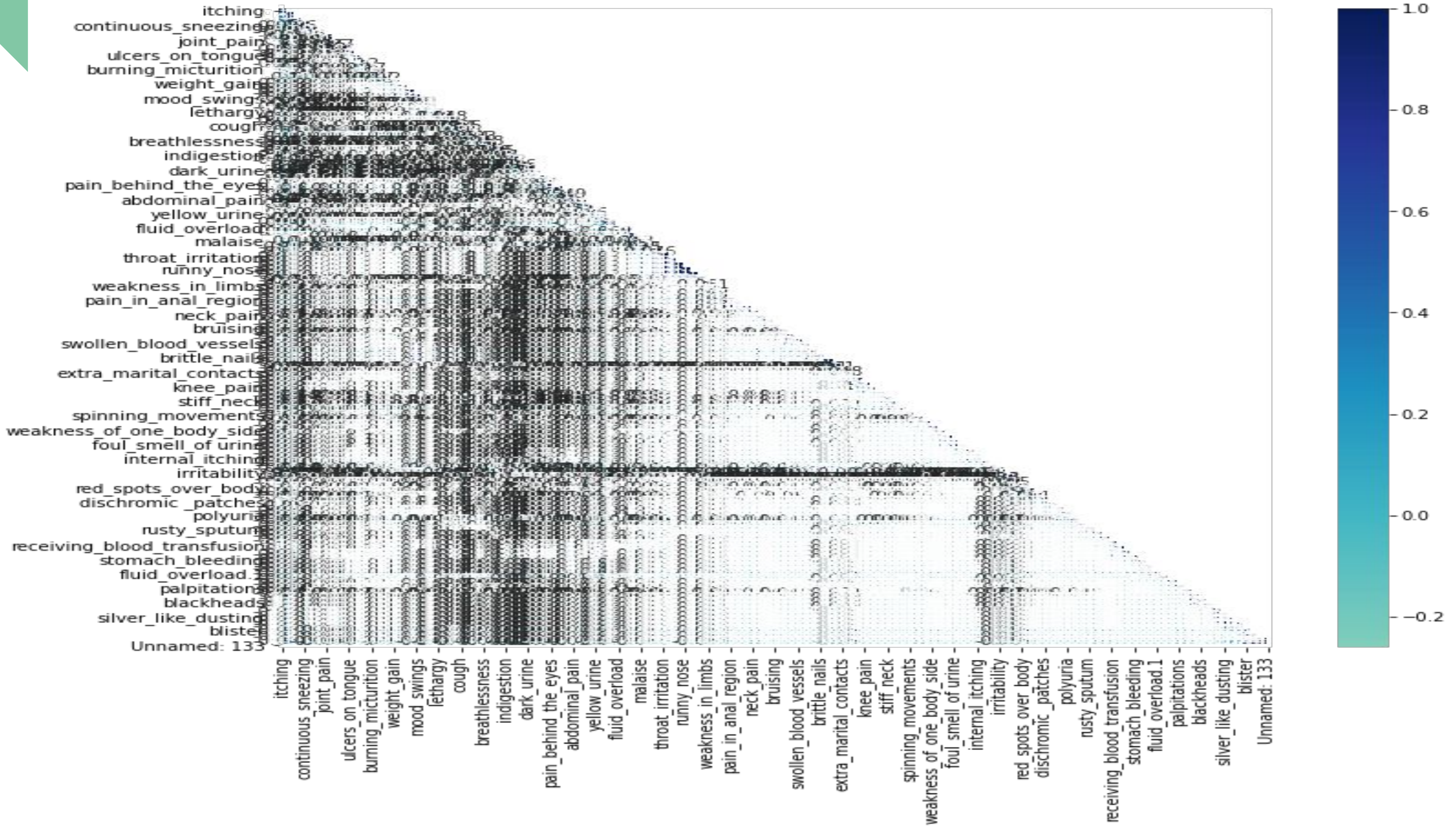
- KNN is a non-parametric algorithm, which means it does not make any assumptions about the underlying data distribution.
- K in K-Nearest Neighbors refers to the number of neighbors that one should take into consideration when predicting the class of a new point.
- KNN is an instance-based algorithm, which means it stores all instances of the training data and uses them to make predictions on new instances.
- The algorithm works by finding the K nearest neighbors to a new instance in the training set based on some distance metric.
- To classify a new instance, KNN takes a majority vote among the K nearest neighbors, where each neighbor's vote is weighted according to its distance from the new instance.
- The value of K is a hyperparameter that can be tuned to improve the performance of the algorithm. Low K - sensitive to outliers .
- KNN can handle both numerical and categorical data, making it a versatile algorithm for a wide range of applications.



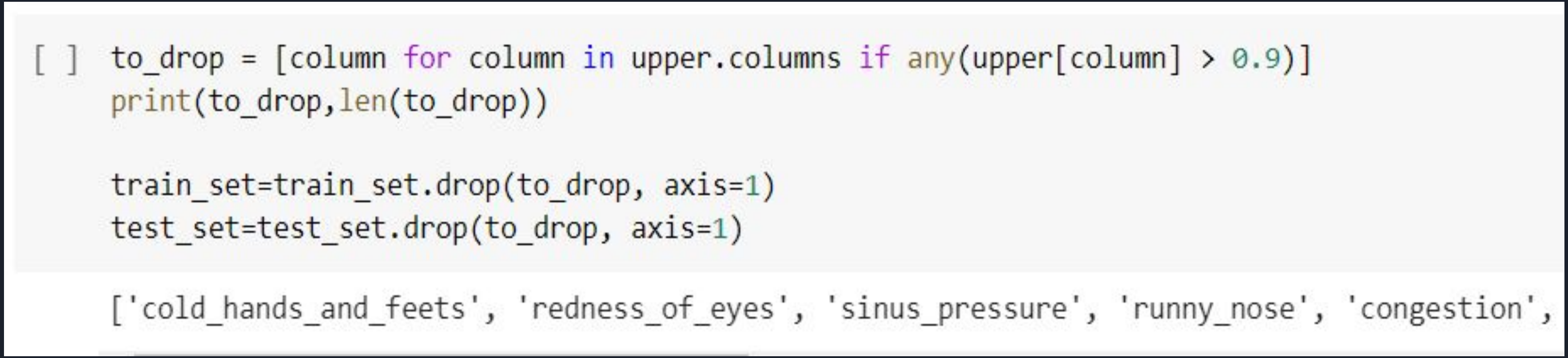
CODE SNIPPETS



CODE SNIPPETS

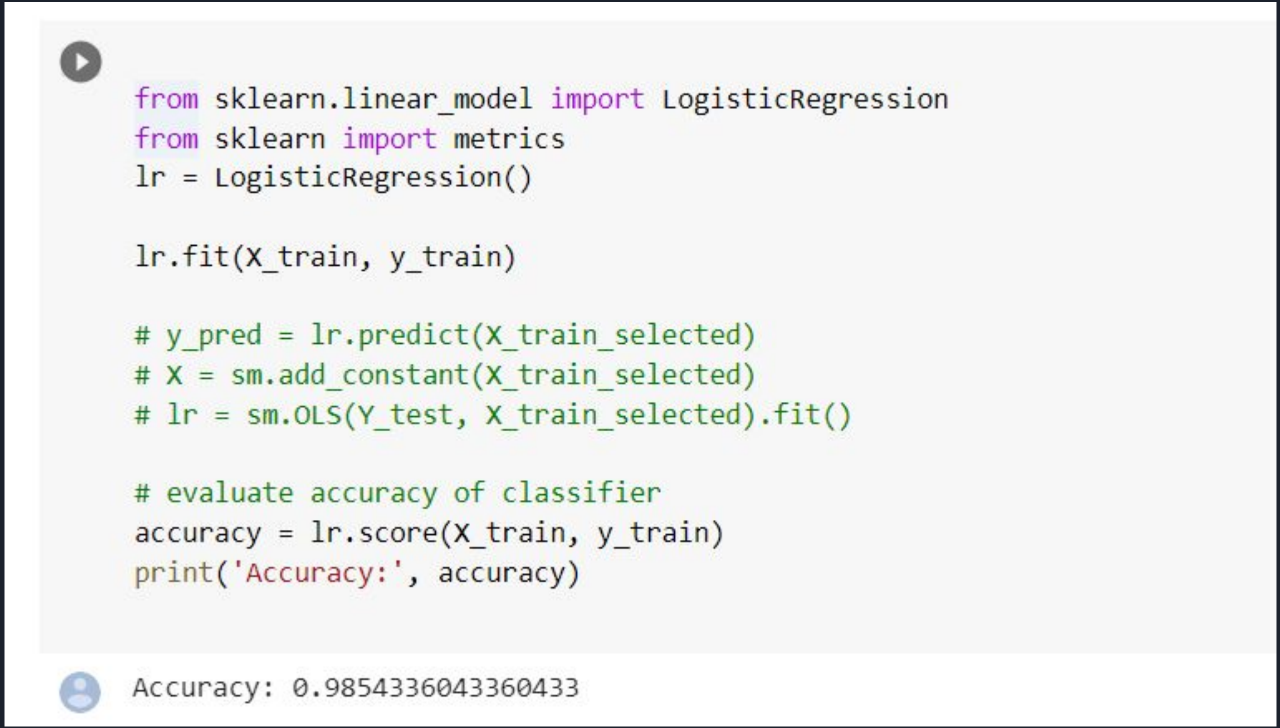


Multicollinearity Removed



```
[ ]  to_drop = [column for column in upper.columns if any(upper[column] > 0.9)]
     print(to_drop,len(to_drop))

     train_set=train_set.drop(to_drop, axis=1)
     test_set=test_set.drop(to_drop, axis=1)

     ['cold_hands_and_feets', 'redness_of_eyes', 'sinus_pressure', 'runny_nose', 'congestion',
```

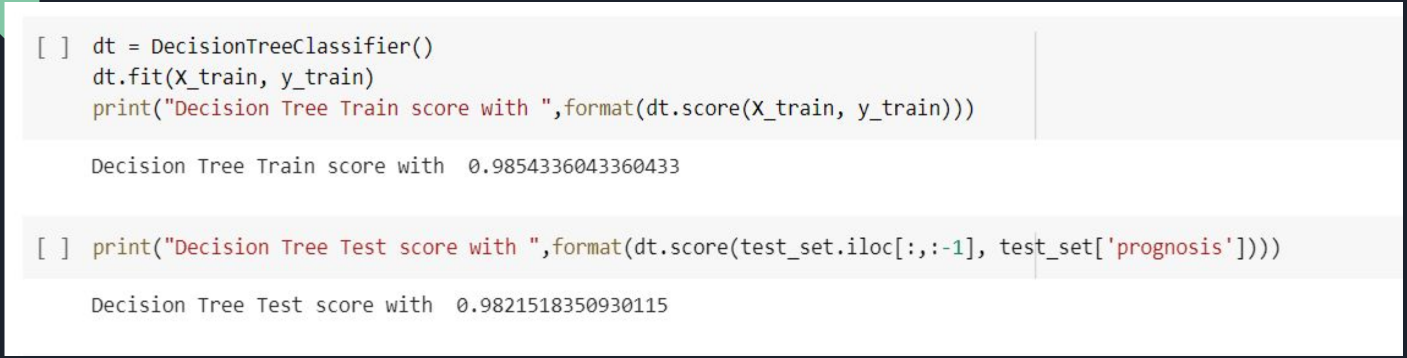
Logistic Regression



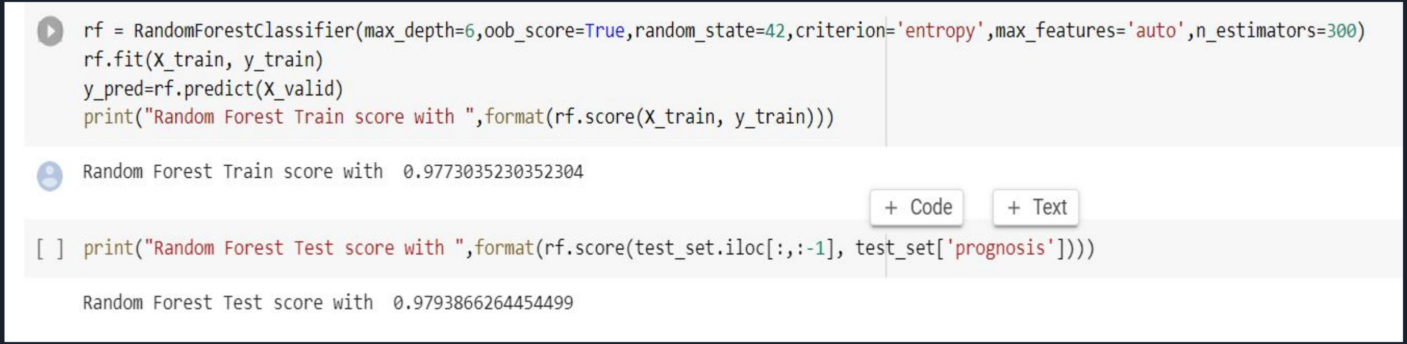
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
lr = LogisticRegression()

lr.fit(X_train, y_train)

# y_pred = lr.predict(X_train_selected)
# X = sm.add_constant(X_train_selected)
# lr = sm.OLS(Y_test, X_train_selected).fit()

# evaluate accuracy of classifier
accuracy = lr.score(X_train, y_train)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.9854336043360433
```

Decision Tree and Random forest

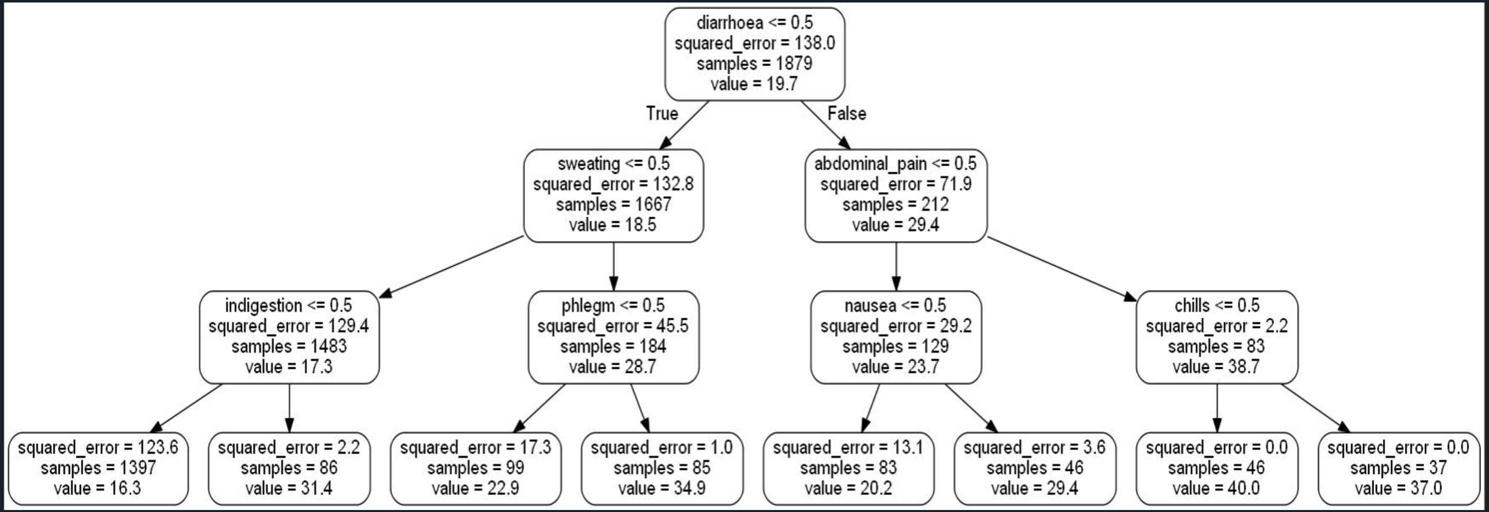


```
[ ]  dt = DecisionTreeClassifier()
     dt.fit(X_train, y_train)
     print("Decision Tree Train score with ",format(dt.score(X_train, y_train)))

     Decision Tree Train score with  0.9854336043360433

[ ]  print("Decision Tree Test score with ",format(dt.score(test_set.iloc[:,:-1], test_set['prognosis'])))

     Decision Tree Test score with  0.9821518350930115
```

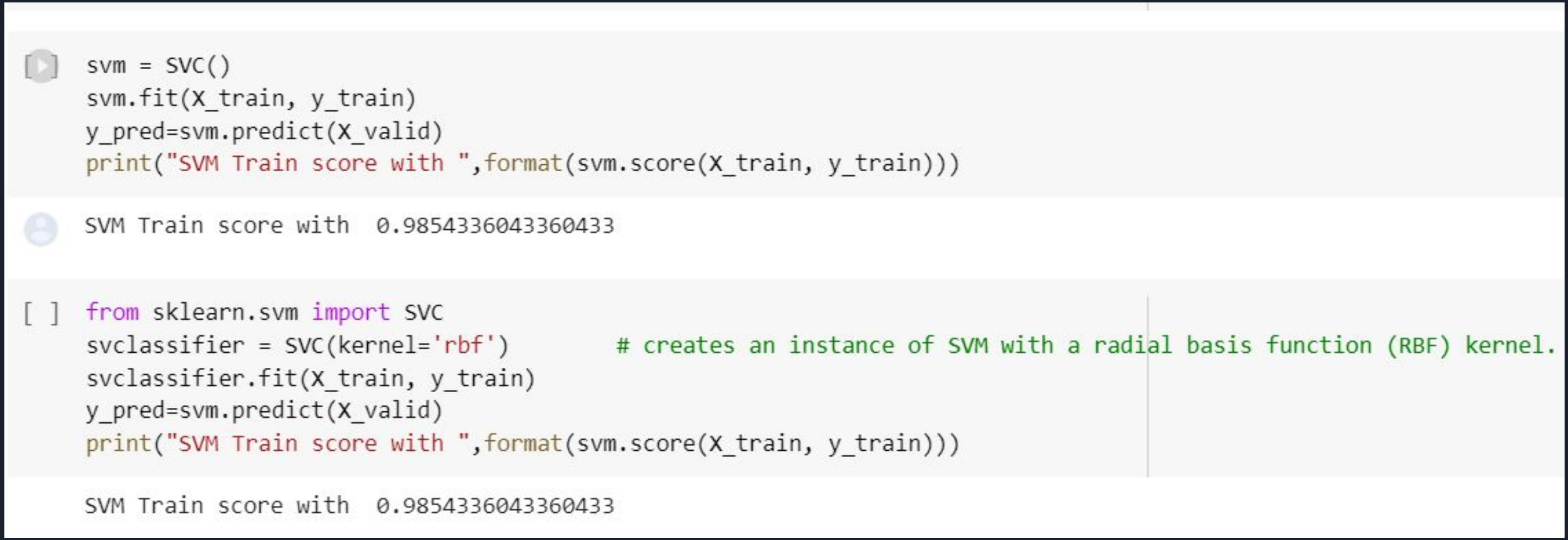


```
rf = RandomForestClassifier(max_depth=6,oob_score=True,random_state=42,criterion='entropy',max_features='auto',n_estimators=300)
rf.fit(X_train, y_train)
y_pred=rf.predict(X_valid)
print("Random Forest Train score with ",format(rf.score(X_train, y_train)))

Random Forest Train score with  0.9773035230352304

[ ]  print("Random Forest Test score with ",format(rf.score(test_set.iloc[:,:-1], test_set['prognosis'])))

     Random Forest Test score with  0.9793866264454499
```

Decision Tree



SVM

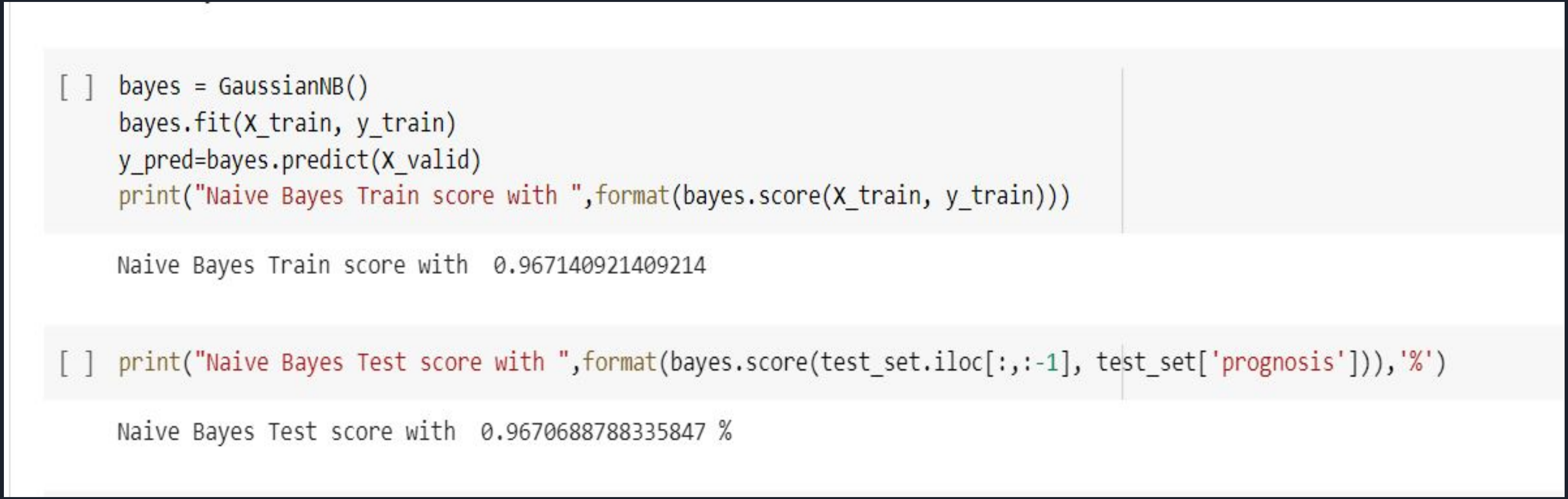

```
svm = SVC()
svm.fit(X_train, y_train)
y_pred=svm.predict(X_valid)
print("SVM Train score with ",format(svm.score(X_train, y_train)))
```

```
SVM Train score with  0.9854336043360433
```

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')        # creates an instance of SVM with a radial basis function (RBF) kernel.
svclassifier.fit(X_train, y_train)
y_pred=svm.predict(X_valid)
print("SVM Train score with ",format(svm.score(X_train, y_train)))
```

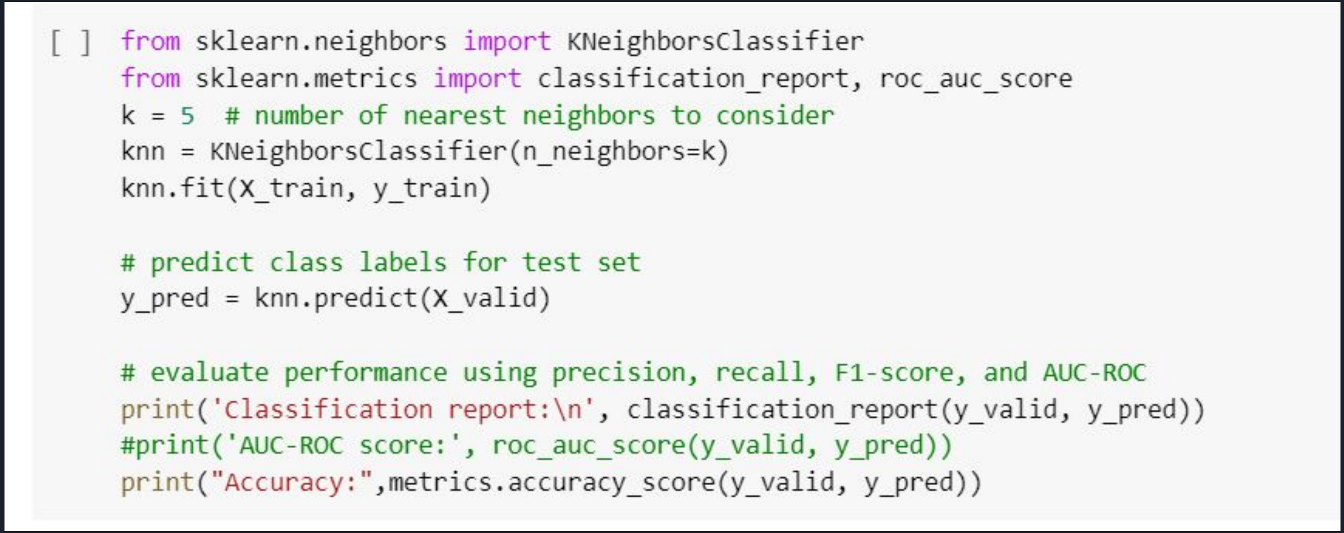
```
SVM Train score with  0.9854336043360433
```

Naive Bayes

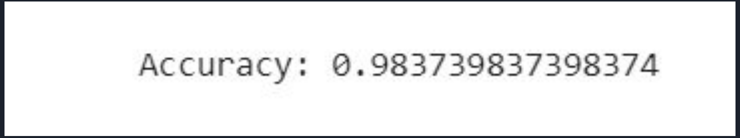


```
[ ]  bayes = GaussianNB()
     bayes.fit(X_train, y_train)
     y_pred=bayes.predict(X_valid)
     print("Naive Bayes Train score with ",format(bayes.score(X_train, y_train)))

     Naive Bayes Train score with  0.967140921409214

[ ]  print("Naive Bayes Test score with ",format(bayes.score(test_set.iloc[:,:-1], test_set['prognosis'])),'%')

     Naive Bayes Test score with  0.9670688788335847 %
```

KNN

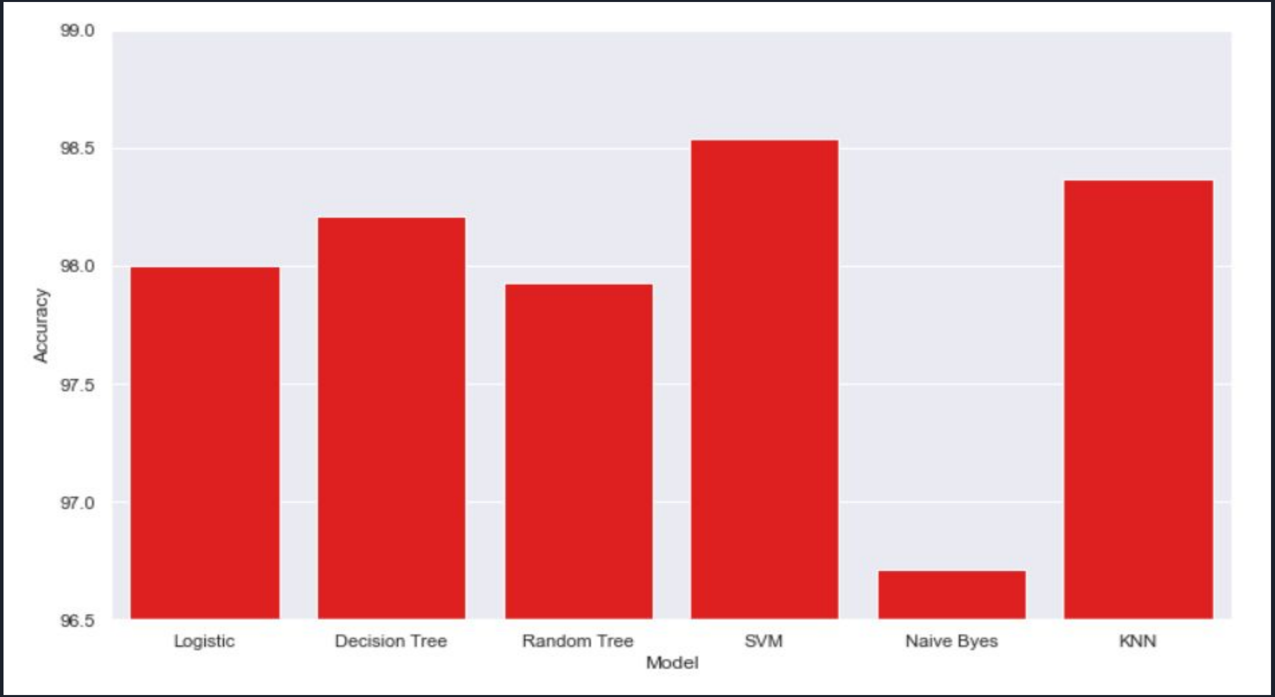


```
[ ] from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import classification_report, roc_auc_score
    k = 5  # number of nearest neighbors to consider
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # predict class labels for test set
    y_pred = knn.predict(X_valid)

    # evaluate performance using precision, recall, F1-score, and AUC-ROC
    print('Classification report:\n', classification_report(y_valid, y_pred))
    #print('AUC-ROC score:', roc_auc_score(y_valid, y_pred))
    print("Accuracy:",metrics.accuracy_score(y_valid, y_pred))
```



```
Accuracy: 0.983739837398374
```

PERFORMANCE COMPARISON



CONCLUSION

Machine learning algorithms offer powerful tools for predicting diseases and improving healthcare outcomes. We explored five different algorithms in this presentation: Support Vector Machines, Naive Bayes, and K-Nearest Neighbors, Decision Tree, Logistic Regression.

By training and testing different algorithms on a dataset, we were able to develop a predictive model that accurately identifies individuals at risk of developing certain diseases.

Overall, each algorithm has its own strengths and weaknesses, and the choice of algorithm will depend on the specific problem at hand. By exploring these different approaches, we can better understand the trade-offs between them and make more informed decisions in healthcare applications.