Disease Prediction from Symptoms

Problem definition:

Objective:

To develop a data mining application to predict disease using symptom data i.e. prognosis.

To develop this application, we use the Columbia University dataset that was available online, the link for which is given in the following section. We start by cleaning the data, then visualizing the data before building a model using both multinomial Naive-Bayes and decision tree algorithms to predict the disease given the symptoms faced by a person.

Data Pre-processing

Original Dataset

The original dataset was taken from the following link: http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

This dataset is a knowledge database of disease-symptom associations generated by an automated method based on information in textual discharge summaries of patients at New York-Presbyterian Hospital admitted during 2004. The first column shows the disease, the second the number of discharge summaries containing a positive and current mention of the disease, and the associated symptom.

Disease	Count of Disease Occurrence	Symptom
UMLS:C0020538_hypertensive disease	3363	UMLS:C0008031_pain chest
		UMLS:C0392680_shortness of breath
		UMLS:C0012833_dizziness
		UMLS:C0004093_asthenia
		UMLS:C0085639_fall
		UMLS:C0039070_syncope
		UMLS:C0042571_vertigo
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0030252_palpitation
		UMLS:C0027497_nausea
		UMLS:C0002962_angina pectoris
		UMLS:C0438716_pressure chest
UMLS:C0011847_diabetes	1421	UMLS:C0032617_polyuria
		UMLS:C0085602_polydypsia
		UMLS:C0392680_shortness of breath
		UMLS:C0008031_pain chest
		UMLS:C0004093_asthenia
		UMLS:C0027497_nausea
		UMLS:C0085619_orthopnea
		UMLS:C0034642_rale
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0241526_unresponsiveness
		UMLS:C0856054_mental status changes
		UMLS:C0042571_vertigo
		UMLS:C0042963_vomiting
		UMLS:C0553668_labored breathing
UMLS:C0011570_depression mental^UMLS:C0011581_depressive disorder	1337	UMLS:C0424000_feeling suicidal
		UMLS:C0438696_suicidal
		UMLS:C0233762_hallucinations auditory
		UMLS:C0150041_feeling hopeless
		UMLS:C0424109_weepiness
		UMLS:C0917801_sleeplessness
		UMLS:C0424230_motor retardation
		UMLS:C0022107_irritable mood

The extraction was performed by copying the data on the website in the .html format and saving it in an Excel file for performing further operations.

Basic cleaning, segmentation of columns and string formatting were performed in excel. The excel sheet was then added to the Google Colab Notebook.

Data preprocessing operations performed:

1. Some spelling mistakes in the names of diseases or symptoms or their codes was rectified
2. Repetitions in symptoms of a disease with the same name or slightly different name were removed
3. Multiple symptoms in the same row were separated
4. The codes which were given to diseases and symptoms were removed as they were irrelevant for our task
5. A cumulative list of all symptoms was made

6. Each symptom was assigned a Boolean value of 0 or 1 for each disease, according to whether the symptom occurs with the disease or not
7. The corresponding disease i.e. prognosis was added in the last column

Final Dataset

	A	B	C	D	E	F	G	H	I	J	K	L
1	itching	skin_rash	nodal_skin_erupt	continuous_snee	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongu	muscle_wasting	vomiting
2	1	1	1	0	0	0	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0	0	0	0	0
4	1	0	1	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0	0	0	0
7	0	1	1	0	0	0	0	0	0	0	0	0
8	1	0	1	0	0	0	0	0	0	0	0	0
9	1	1	0	0	0	0	0	0	0	0	0	0
10	1	1	1	0	0	0	0	0	0	0	0	0
11	1	1	1	0	0	0	0	0	0	0	0	0
12	0	0	0	1	1	1	0	0	0	0	0	0
13	0	0	0	0	1	1	0	0	0	0	0	0
14	0	0	0	1	0	1	0	0	0	0	0	0
15	0	0	0	1	1	0	0	0	0	0	0	0
16	0	0	0	1	1	1	0	0	0	0	0	0
17	0	0	0	0	1	1	0	0	0	0	0	0
18	0	0	0	1	0	1	0	0	0	0	0	0
19	0	0	0	1	1	0	0	0	0	0	0	0
20	0	0	0	1	1	1	0	0	0	0	0	0

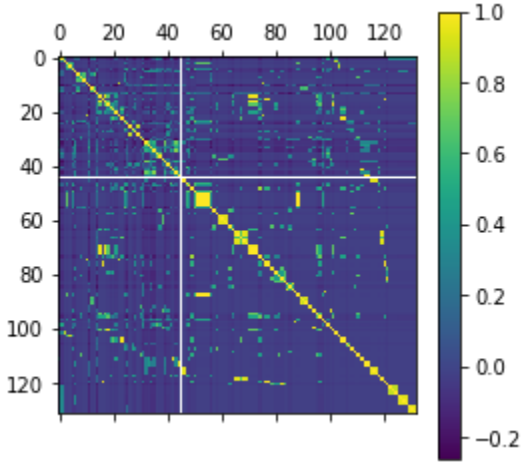
Columns 1 to 12

blackheads	scurring	skin_peeling	silver_like_dustir	small_dents_in_r	inflammatory_na	blister	red_sore_around	yellow_crust_ooz	prognosis
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Fungal infection
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy
0	0	0	0	0	0	0	0	0	Allergy

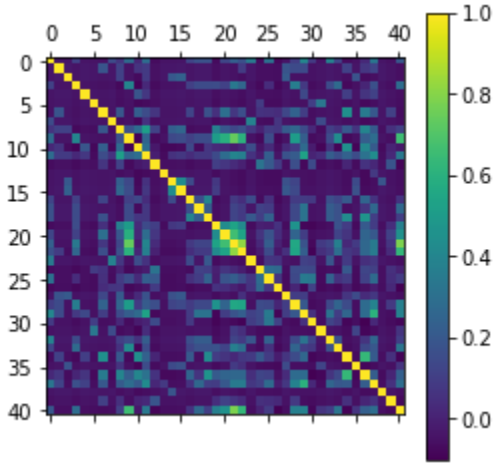
Columns 123 to 133

Data Visualisations

Correlation heatmap for relationship between the symptoms:



Correlation heatmap for relationship between the diseases:



Algorithms & Implementation

Implementation link:
https://colab.research.google.com/drive/1gS8L3QydlmP3DuNXhJkYdarOscCj7frF

We have used 2 algorithms for this dataset so that we could also know which one yields better results.

Algorithms used:

1. Multinomial Naive Bayes Classifier.
2. Decision Tree

Multinomial Naive Bayes Classifier

MultinomialNB implements the naive Bayes algorithm for multinomial distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta y=(\theta y1,\ldots,\theta yn)$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θyi is the probability $P(xi|y)$ of feature i appearing in a sample belonging to class y The parameters θy is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$\theta \hat{}yi=Nyi+\alpha Ny+\alpha n$

where $Nyi=\sum x\in Txi$ is the number of times feature i appears in a sample of class y in the training set T, and $Ny=\sum i=lnNyi$ is the total count of all features for class Y

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha=1$ is called Laplace smoothing, while $\alpha<1$ is called Lidstone smoothing.

Decision Tree description

Decision trees are the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal

node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
Construction of Decision Tree: A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general, decision tree classifiers have good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Implementation outputs:

1). Multinomial Naive Bayes Classifier results:

- Training results

```
[ ]  mnb.score(x_test, y_test)
```

```
1.0
```

- Validation results

```
from sklearn import model_selection
print ("cross result========")
scores = model_selection.cross_val_score(mnb, x_test, y_test, cv=3)
print (scores)
print (scores.mean())
```

```
cross result========
[1. 1. 1.]
1.0
```

- Test results

```
] mnb.score(testx, testy)
```

```
0.926829268292683
```

2). Decision Tree results:

- Training results

```
print ("DecisionTree")
dt = DecisionTreeClassifier(min_samples_split=20)
clf_dt=dt.fit(x_train,y_train)
print ("Acurracy: ", clf_dt.score(x_test,y_test))
```

```
DecisionTree
Acurracy:  0.9772167487684729
```

- Validation results

```
from sklearn import model_selection
print ("cross result========")
scores = model_selection.cross_val_score(dt, x_test, y_test, cv=3)
print (scores)
print (scores.mean())
```

```
cross result========
[0.95503597 0.95563771 0.93927894]
0.9499842055175566
```

- Test results

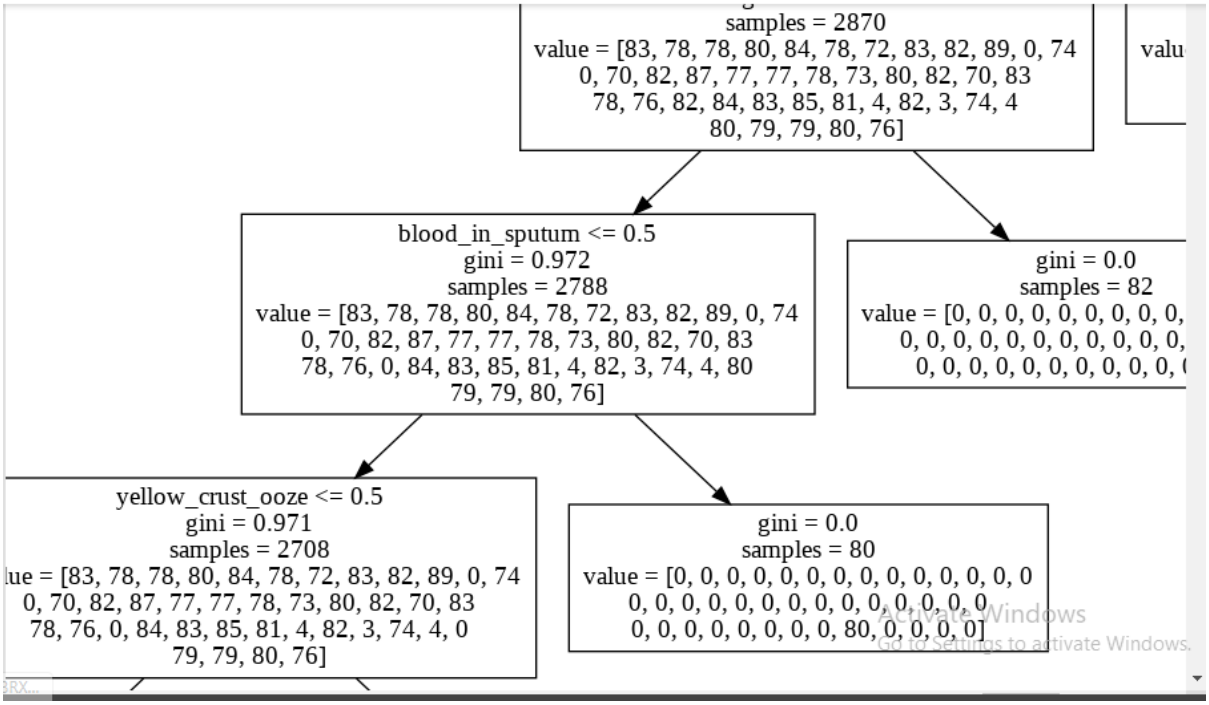
```
print ("Acurracy on the actual test data: ", clf_dt.score(testx,testy))
```

```
Acurracy on the actual test data:  0.926829268292683
```

Decision Tree

The entire decision tree is too big to be inserted here, so only a part of it is shown here. The entire image can be found on this link:

https://drive.google.com/file/d/1bJ6ZD5ZDh0Cqs0R23DWBEHPYP6flxLx9/view?usp=sharing



Evaluation & results:

After running the two algorithms on our training and testing sets we found the accuracy scores as shown above. We now focus on interpreting how important the model thinks each symptom(input) is. This analysis uses a factor called importance which is a measure of the contribution of the input of the variable to the output predicted. This can be described as follows:

The importance metric is associated with each and every node in the decision tree. It measures the decrease in the impurity weighted by weighted inputs. The contribution of the children is then removed from that of the contribution of the parent. We thus obtain the following formula. This is the method used in common implementations like sklearn:

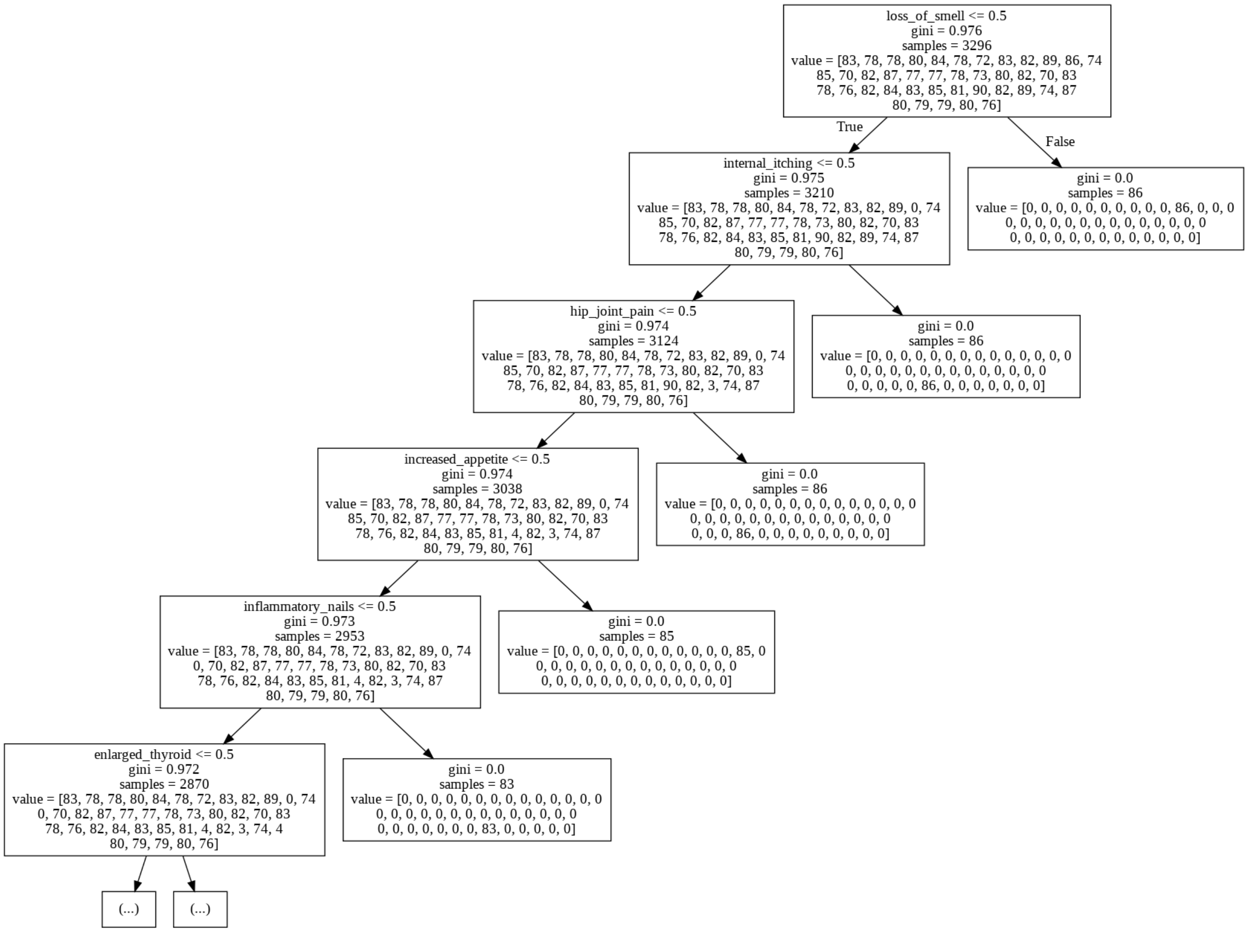
$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

where:
ni sub(j)= the importance of node j
w sub(j) = weighted number of samples reaching node j
C sub(j)= the impurity value of node j
left(j) = child node from left split on node j
right(j) = child node from right split on node j

We thus obtain the following results with decision tree.

These are the top 20 symptoms with their importance in brackets:
1. feature 88 - loss_of_smell (0.026973)
2. feature 93 - internal_itching (0.026919)
3. feature 79 - hip_joint_pain (0.026895)
4. feature 104 - increased_appetite (0.026657)
5. feature 48 - malaise (0.026156)
6. feature 128 - inflammatory_nails (0.025972)
7. feature 71 - enlarged_thyroid (0.025737)
8. feature 118 - blood_in_sputum (0.025126)
9. feature 131 - yellow_crust_ooze (0.025049)
10. feature 2 - nodal_skin_eruptions (0.024985)
11. feature 86 - unsteadiness (0.024733)
12. feature 96 - irritability (0.024652)
13. feature 19 - weight_loss (0.024579)
14. feature 119 - prominent_veins_on_calf (0.024460)
15. feature 117 - fluid_overload.1 (0.024457)
16. feature 95 - depression (0.024430)
17. feature 109 - lack_of_concentration (0.024375)
18. feature 97 - muscle_pain (0.024292)
19. feature 41 - mild_fever (0.024262)
20. feature 63 - neck_pain (0.024226)

The same can also be seen by ordering the inputs according to the depth of the tree. The top 5 nodes can be seen as following:



Fooling a decision tree model by misusing the symptoms and altering the tree traversal.

Example:

```
symptoms = ['skin_rash','itching','nodal_skin_eruptions','increased_appetite','irritability']
ipt = [0 for i in range(len(features))]
for s in symptoms:
  ipt[cols.index(s)]=1
ipt = np.array([ipt])
print(ipt)
print(dt.predict(ipt))
dt.predict_proba(ipt)
```

```
[[1 1 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 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 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 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]]
['Diabetes ']
array([[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.]])
```

Conclusion:

After the data gathering and pre-processing or cleaning the data, we created a Machine Learning model using algorithms like Naive Bayes, Decision tree, that is able to correctly predict the type of disease, given the different type of symptoms experienced by the user.

The decision tree approach proves to be better as naive Bayes assumes that all the features are independent of one another and gives the probability whereas in our dataset multiple symptoms together can accurately tell us about the disease.