```
import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
from sklearn.svm import SVR
from matplotlib.colors import ListedColormap
```

```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import pandas as pd

test = pd.read_csv("Testing.csv")
train = pd.read_csv("Training.csv")
```

```
train.sample(n=1)
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_or
4110	0	0	0	0	0	0	0	0	0	

1 rows × 133 columns

```
data = pd.concat([train, test])
```

```
data.sample(10)
#This will give us randome sample data
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_or
2307	0	0	0	0	0	1	0	0	0	
711	0	0	0	0	0	0	0	0	0	
2373	0	0	0	0	0	0	0	0	0	
781	0	1	0	0	0	0	0	0	0	
1785	1	0	0	0	0	0	0	0	0	
4321	1	1	0	0	0	0	0	0	0	
4003	0	0	0	1	0	1	0	0	0	
1994	0	0	0	0	0	0	0	0	0	
3074	0	1	0	0	0	0	0	0	0	
1816	0	1	0	0	0	1	0	0	0	

10 rows × 133 columns

```
data.head(5)
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_to
0	1	1	1	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	
2	1	0	1	0	0	0	0	0	0	
3	1	1	0	0	0	0	0	0	0	
4	1	1	1	0	0	0	0	0	0	

5 rows × 133 columns

```
data.tail(5)
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_t
36	0	0	0	0	0	0	0	0	0	
37	0	1	0	0	0	0	0	0	0	
38	0	0	0	0	0	0	0	0	0	
39	0	1	0	0	0	0	1	0	0	
40	0	1	0	0	0	0	0	0	0	

5 rows × 133 columns

```
data.columns
```

```
Index(['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing',
       'shivering', 'chills', 'joint_pain', 'stomach_pain', 'acidity',
       'ulcers_on_tongue',
       ...
       'blackheads', 'scurring', 'skin_peeling', 'silver_like_dusting',
       'small_dents_in_nails', 'inflammatory_nails', 'blister',
       'red_sore_around_nose', 'yellow_crust_ooze', 'prognosis'],
      dtype='object', length=133)
```

```
data.shape
```

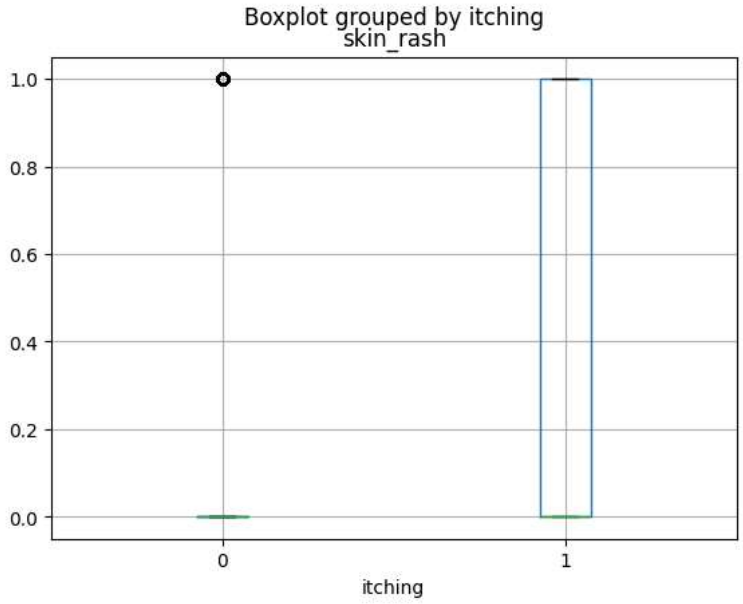
```
(4961, 133)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4961 entries, 0 to 40
Columns: 133 entries, itching to prognosis
dtypes: int64(132), object(1)
memory usage: 5.1+ MB
```

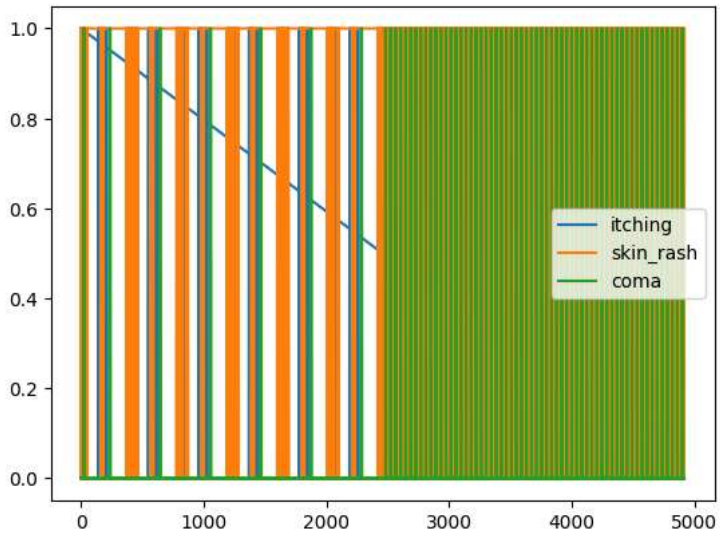
```
# What we are expecting from visualization. ?
data.boxplot(column ='skin_rash', by='itching') #boxplot shows outlier, median,Q3,Q1
```

```
<Axes: title={'center': 'skin_rash'}, xlabel='itching'>
```



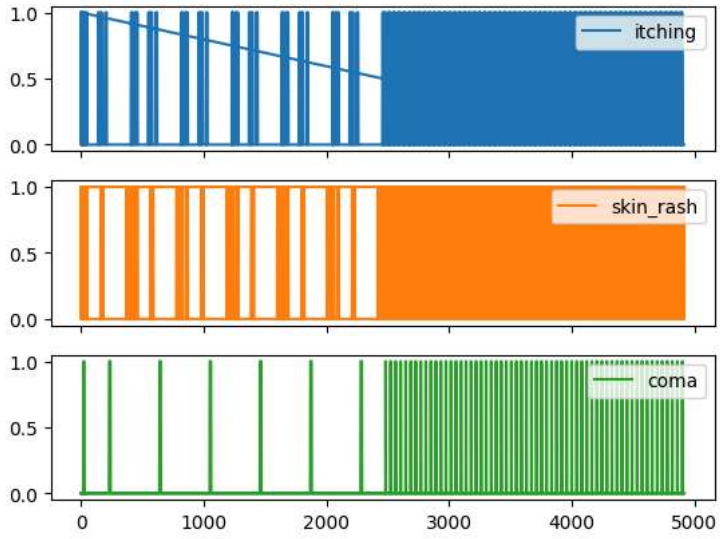

```
# # What we are expecting from visualization. ?

data1 =data.loc[:,["itching","skin_rash","coma"] ]
data1.plot()
```

```
<Axes: >
```



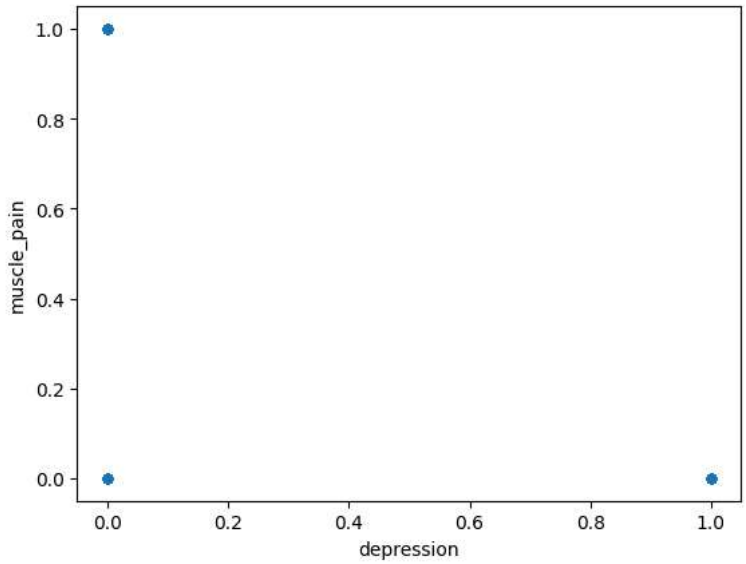

```
## What we are expecting from visualization. ?

data1.plot(subplots =True)
```

```
array([<Axes: >, <Axes: >, <Axes: >], dtype=object)
```



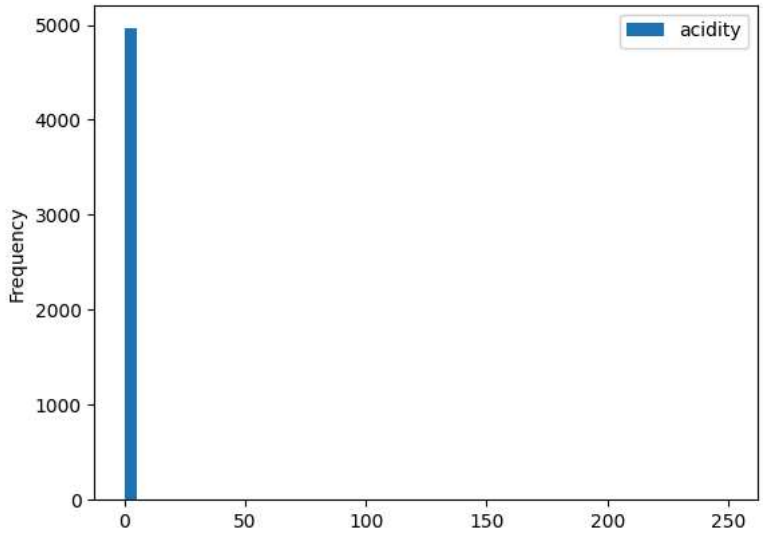

```
data.plot(kind= "scatter",x="depression", y="muscle_pain")
```

```
<Axes: xlabel='depression', ylabel='muscle_pain'>
```




```
data.plot(kind="hist", y="acidity", bins = 50, range=(0,250))
```

```
<Axes: ylabel='Frequency'>
```



```
#Importing the train_test_split functionality
from sklearn.model_selection import train_test_split

X, y=data.iloc[:,:-1], data.iloc[:,-1]

#Spliting the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)
#70% training and 30% test
```

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

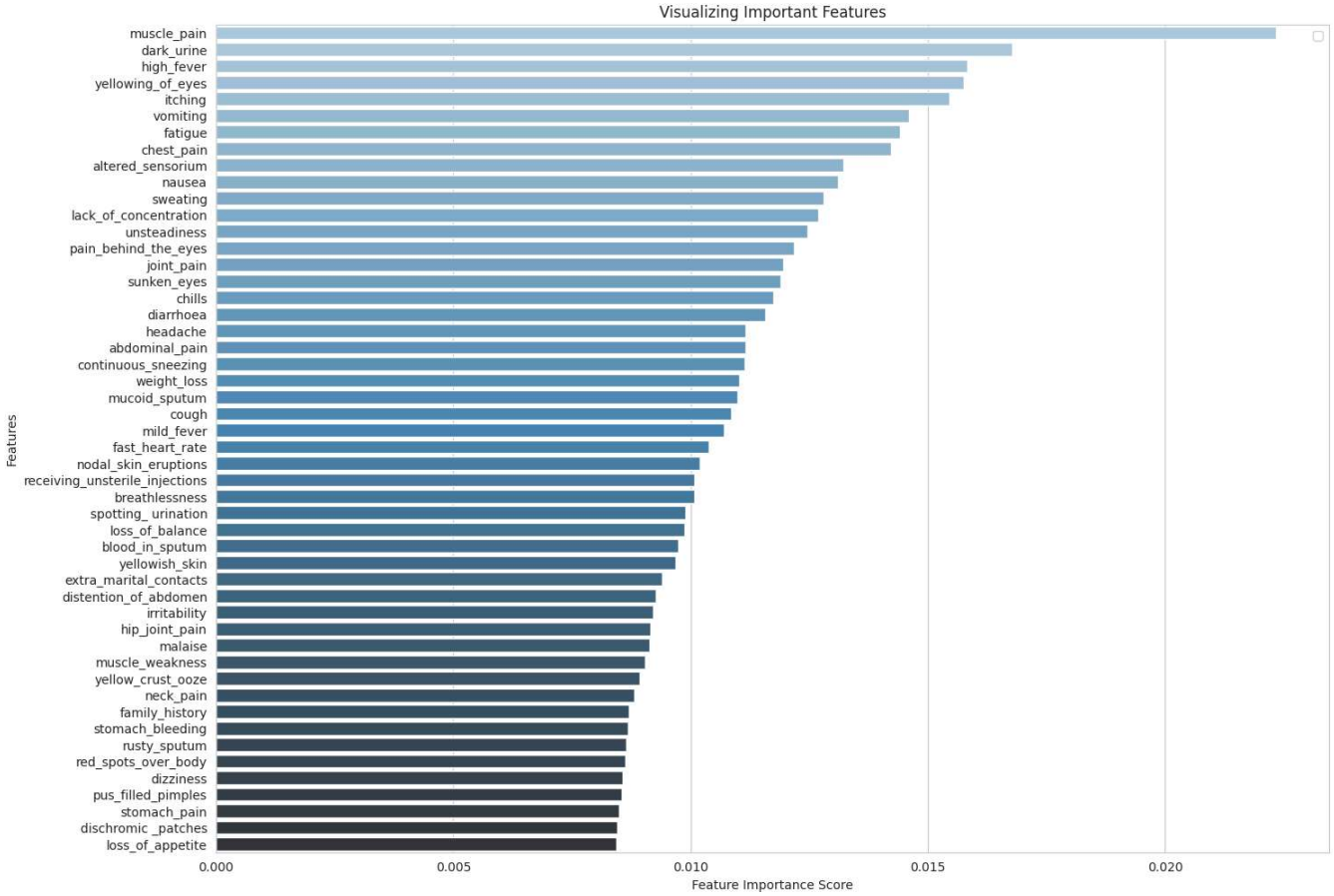
```
Accuracy: 1.0
```

```
feature_imp[::-1].index
```

```
Index(['loss_of_appetite', 'dischromic _patches', 'stomach_pain',
       'pus_filled_pimples', 'dizziness', 'red_spots_over_body',
       'rusty_sputum', 'stomach_bleeding', 'family_history', 'neck_pain',
       'yellow_crust_ooze', 'muscle_weakness', 'malaise', 'hip_joint_pain',
       'irritability', 'distention_of_abdomen', 'extra_marital_contacts',
       'yellowish_skin', 'blood_in_sputum', 'loss_of_balance',
       'spotting_ urination', 'breathlessness',
       'receiving_unsterile_injections', 'nodal_skin_eruptions',
       'fast_heart_rate', 'mild_fever', 'cough', 'mucoid_sputum',
       'weight_loss', 'continuous_sneezing', 'abdominal_pain', 'headache',
       'diarrhoea', 'chills', 'sunken_eyes', 'joint_pain',
       'pain_behind_the_eyes', 'unsteadiness', 'lack_of_concentration',
       'sweating', 'nausea', 'altered_sensorium', 'chest_pain', 'fatigue',
       'vomiting', 'itching', 'yellowing_of_eyes', 'high_fever', 'dark_urine',
       'muscle_pain'],
      dtype='object')
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

fig = plt.gcf()
fig.set_size_inches(16, 12)

# Creating a bar plot
sns.set_style("whitegrid")
sns.barplot(x=feature_imp, y=feature_imp.index, palette='Blues_d',color='white')

# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
/tmp/ipython-input-1280776881.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `l

  sns.barplot(x=feature_imp, y=feature_imp.index, palette='Blues_d',color='white')
/tmp/ipython-input-1280776881.py:16: UserWarning: No artists with labels found to put in legend.  Note that artists whose label sta
  plt.legend()
```




```
X_reduced, y = data[['receiving_blood_transfusion', 'red_sore_around_nose',
       'abnormal_menstruation', 'continuous_sneezing', 'breathlessness',
       'blackheads', 'shivering', 'dizziness', 'back_pain', 'unsteadiness',
       'yellow_crust_ooze', 'muscle_weakness', 'loss_of_balance', 'chills',
       'ulcers_on_tongue', 'stomach_bleeding', 'lack_of_concentration', 'coma',
       'neck_pain', 'weakness_of_one_body_side', 'diarrhoea',
       'receiving_unsterile_injections', 'headache', 'family_history',
       'fast_heart_rate', 'pain_behind_the_eyes', 'sweating', 'mucoid_sputum',
       'spotting_ urination', 'sunken_eyes', 'dischromic _patches', 'nausea',
       'dehydration', 'loss_of_appetite', 'abdominal_pain', 'stomach_pain',
       'yellowish_skin', 'altered_sensorium', 'chest_pain', 'muscle_wasting',
       'vomiting', 'mild_fever', 'high_fever', 'red_spots_over_body',
       'dark_urine', 'itching', 'yellowing_of_eyes', 'fatigue', 'joint_pain',
       'muscle_pain']], data.iloc[:,-1]

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X_reduced, y, test_size=0.3) # 70% training and 30% test
```

```
#Create a Gaussian Classifier
clf2=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf2.fit(X_train,y_train)

y_pred=clf2.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9449294828744124
```

```
fig = plt.gcf()
fig.set_size_inches(16, 12)

# Creating a bar plot
sns.set_style("whitegrid")
sns.barplot(x=feature_imp2, y=feature_imp2.index, palette='Blues_d',color='white')

# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
/tmp/ipython-input-2959618017.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `l

  sns.barplot(x=feature_imp2, y=feature_imp2.index, palette='Blues_d',color='white')
/tmp/ipython-input-2959618017.py:12: UserWarning: No artists with labels found to put in legend.  Note that artists whose label sta
  plt.legend()
```

Visualizing Important Features

```
#Plotting through bar chart
data['prognosis'].value_counts(normalize = True).plot.bar()
plt.subplots_adjust(left = 0.9, right = 2 , top = 2, bottom = 1)
```