Northeastern University

MACHINE LEARNING IN HEALTHCARE: A HEAD-TO-HEAD COMPARISON OF DISEASE PREDICTION TECHNIQUES

Soham Shah (002703848)

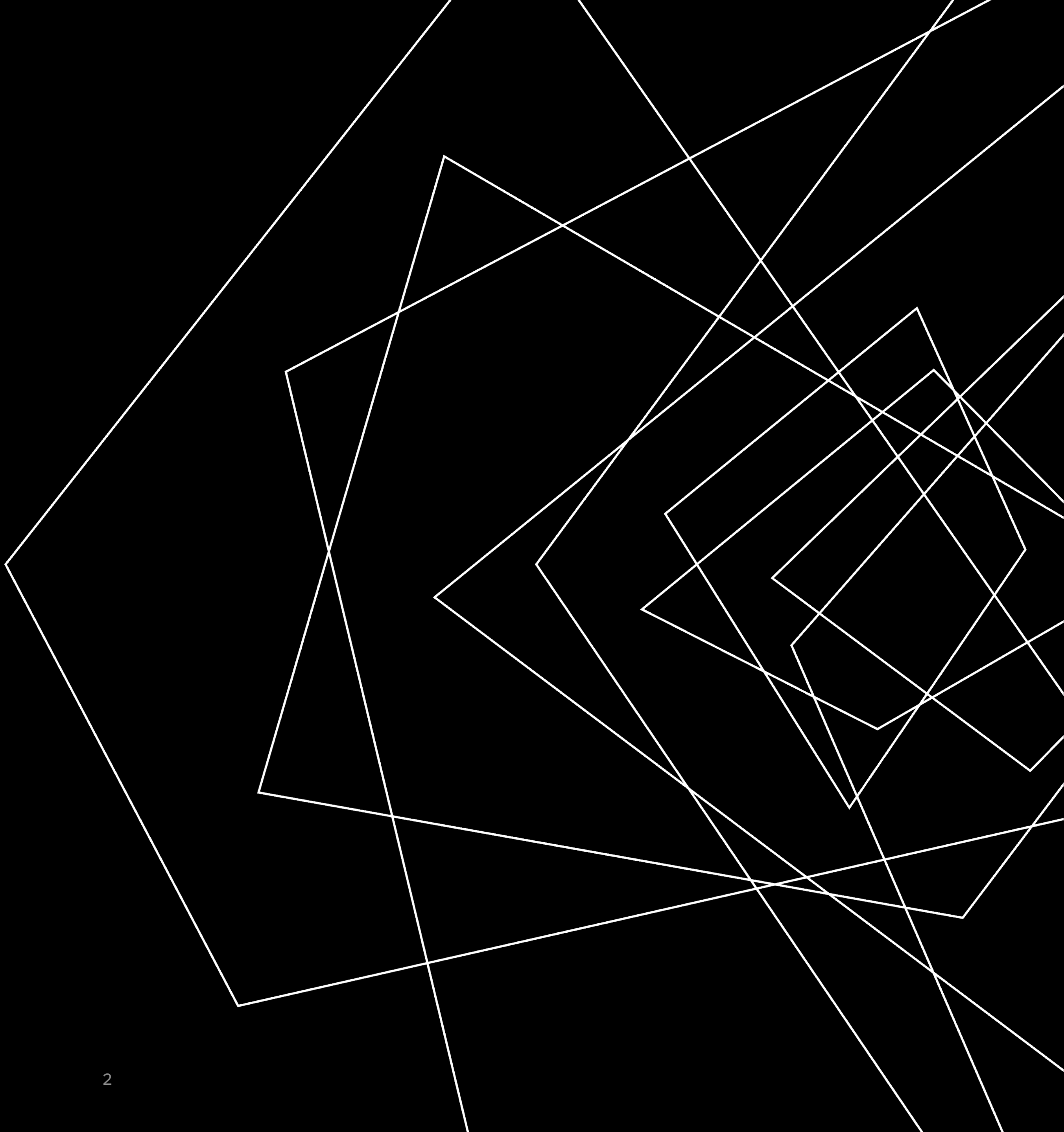
Arundhati Pathrikar (002780632)

Mamtha Shetty (002746925)

Shantanu Pingale (002749482)

AGENDA

- Problem Description
- Approach and Algorithms
- Implementation environment and code example
- Results
- Performance comparison
- Conclusion
- Contribution



PROBLEM STATEMENT

- Dataset comprises of two CSV files containing a total of 133 columns each Out of these columns, 132 represent various symptoms that an individual may exhibit
- The last column denotes the predicted outcome or prognosis of the disease
- These symptoms are mapped to 42 diseases
- We had to train the model on training data and test it on testing data using different classification models.

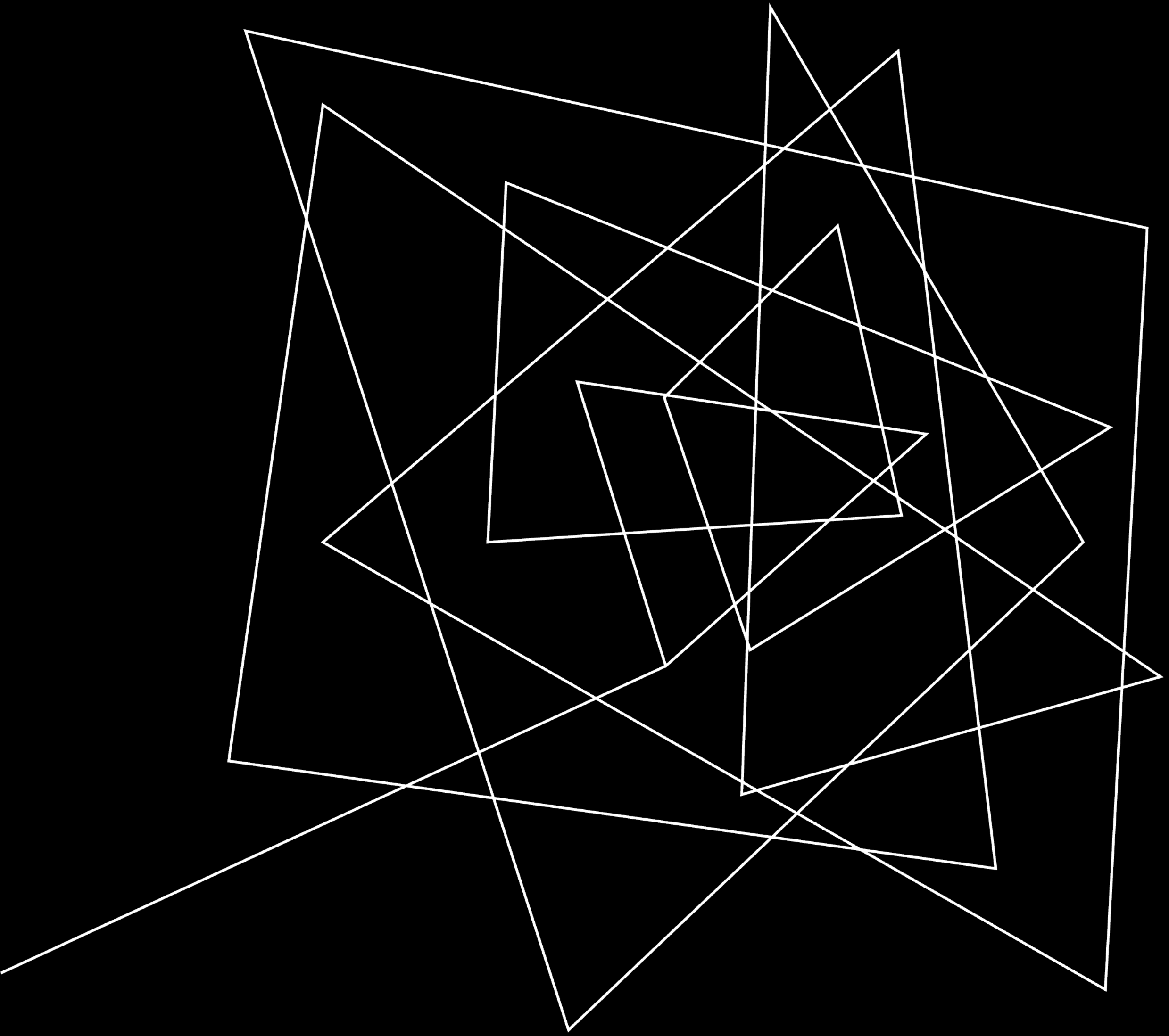
DATA PREPARATION

Removing Inconsistencies

We calculated the mean of each feature because in classification it's either 0 or 1, This also demonstrates that there is no further way to normalize the data

Vomiting and fatigue has a mean value of 0.404761905 and are considered very important characteristics, but when we remove them to see the change in accuracy, the output is different than expected. This means these functions are too general

After that we removed a feature 'fluid_overload' which had value of mean as 0 , trained the model again, to conclude that no effect on the output



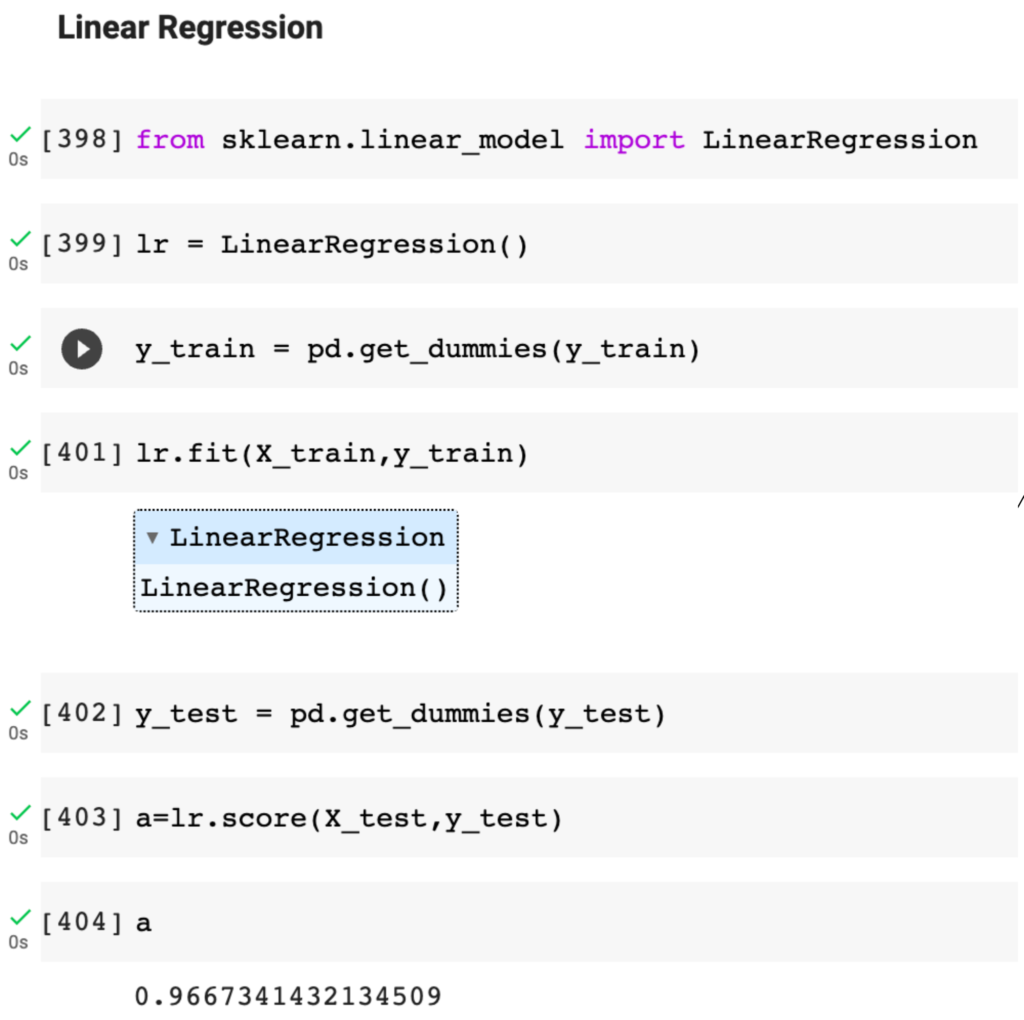
APPROACH AND ALGORITHMS USED

- Linear Regression
- Support Vector Machine (SVM)
- Decision Tree
- Random Forest
- XGBoost

LINEAR REGRESSION

- Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables
- It assumes a linear relationship between the dependent variable and the independent variable(s)
- To predict the accuracy of the Prognosis, we use the Training dataset and Prognosis as the target variable

CODE SNIPPET – LINEAR REGRESSION



Linear Regression

```
[398] from sklearn.linear_model import LinearRegression
```

```
[399] lr = LinearRegression()
```

```
y_train = pd.get_dummies(y_train)
```

```
[401] lr.fit(X_train,y_train)
```

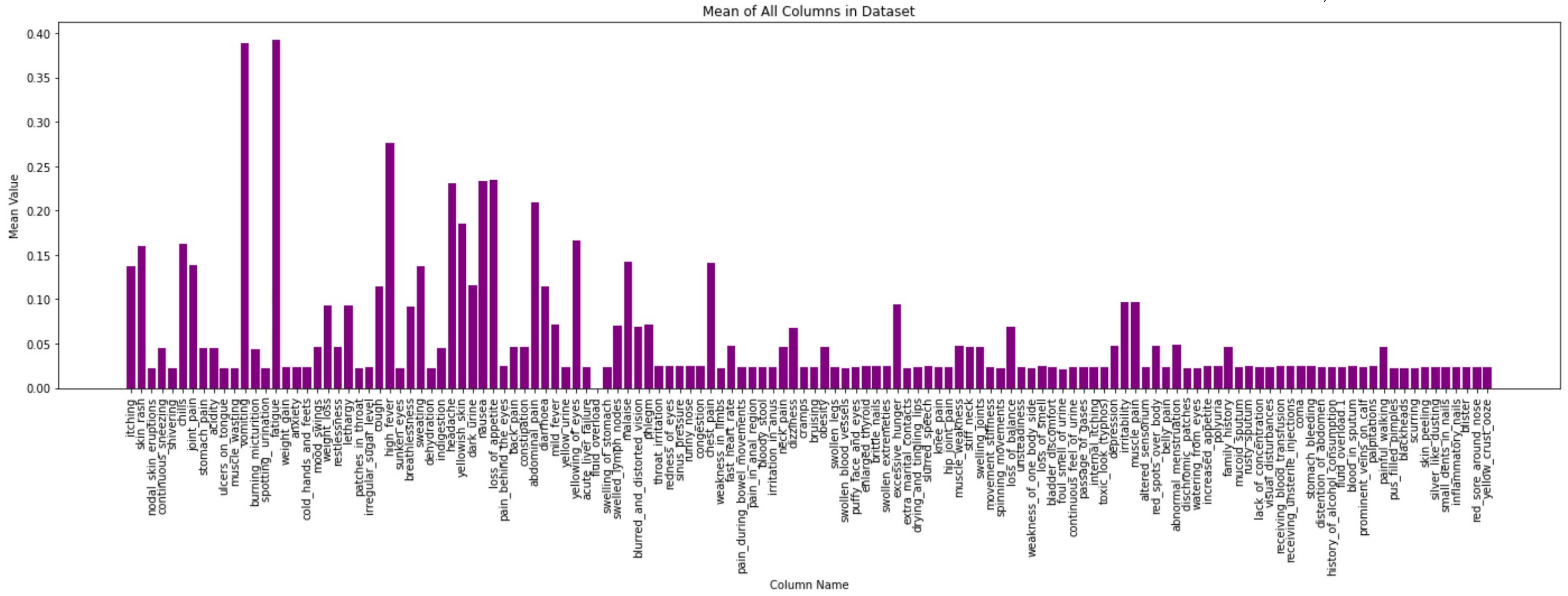
```
▾ LinearRegression
LinearRegression()
```

```
[402] y_test = pd.get_dummies(y_test)
```

```
[403] a=lr.score(X_test,y_test)
```

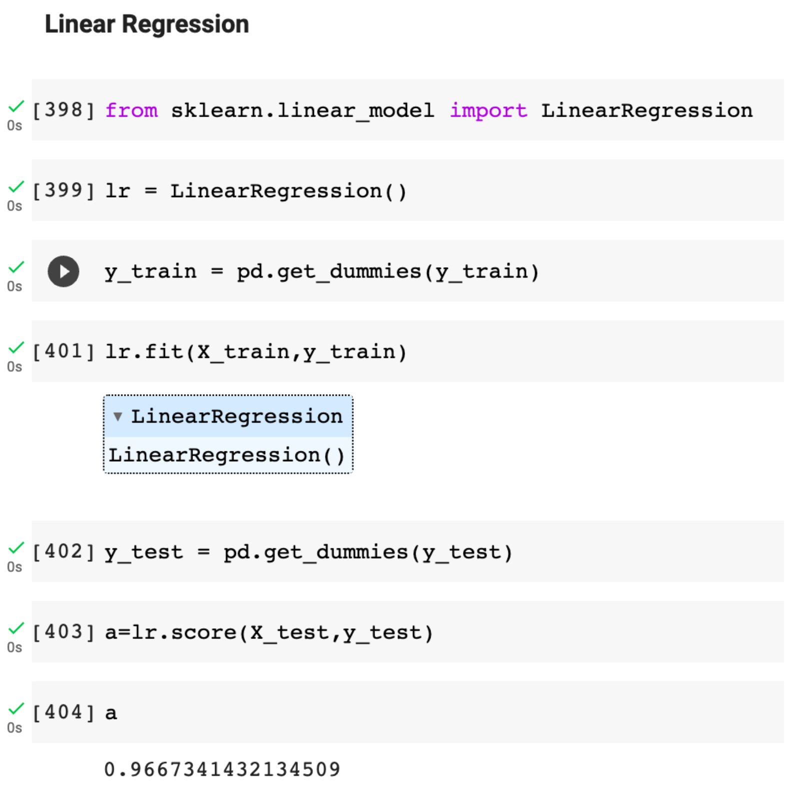
```
[404] a
```

```
0.9667341432134509
```



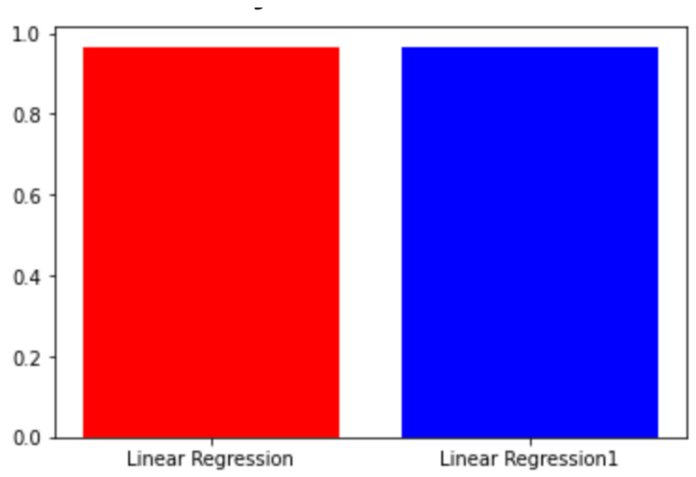
Mean of All Columns in Dataset
Mean Value
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Column Name
itching
skin_rash
nodal_skin_eruptions
continuous_sneezing
shivering
chills
joint_pain
stomach_pain
acidity
ulcers_on_tongue
muscle_wasting
vomiting
burning_micturition
spotting_ urination
fatigue
weight_gain
anxiety
cold_hands_and_feets
mood_swings
weight_loss
restlessness
lethargy
patches_in_throat
irregular_sugar_level
cough
high_fever
sunken_eyes
breathlessness
sweating
dehydration
indigestion
headache
yellowish_skin
dark_urine
nausea
loss_of_appetite
pain_behind_the_eyes
back_pain
constipation
abdominal_pain
diarrhoea
mild_fever
yellow_urine
yellowing_of_eyes
acute_liver_failure
fluid_overload
swelling_of_stomach
swelled_lymph_nodes
malaise
blurred_and_distorted_vision
phlegm
throat_irritation
redness_of_eyes
sinus_pressure
runny_nose
congestion
chest_pain
weakness_in_limbs
fast_heart_rate
pain_during_bowel_movements
pain_in_anal_region
bloody_stool
irritation_in_anus
neck_pain
dizziness
cramps
bruising
obesity
swollen_legs
swollen_blood_vessels
puffy_face_and_eyes
enlarged_thyroid
brittle_nails
swollen_extremeties
excessive_hunger
extra_marital_contacts
drying_and_tingling_lips
slurred_speech
knee_pain
hip_joint_pain
muscle_weakness
stiff_neck
swelling_joints
movement_stiffness
spinning_movements
loss_of_balance
unsteadiness
weakness_of_one_body_side
loss_of_smell
bladder_discomfort
foul_smell_of urine
continuous_feel_of_urine
passage_of_gases
internal_itching
toxic_look_(typhos)
depression
irritability
muscle_pain
altered_sensorium
red_spots_over_body
belly_pain
abnormal_menstruation
dischromic _patches
watering_from_eyes
increased_appetite
polyuria
family_history
mucoid_sputum
rusty_sputum
lack_of_concentration
visual_disturbances
receiving_blood_transfusion
receiving_unsterile_injections
coma
stomach_bleeding
distention_of_abdomen
history_of_alcohol_consumption
fluid_overload.1
blood_in_sputum
prominent_veins_on_calf
palpitations
painful_walking
pus_filled_pimples
blackheads
scurring
skin_peeling
silver_like_dusting
small_dents_in_nails
inflammatory_nails
blister
red_sore_around_nose
yellow_crust_ooze

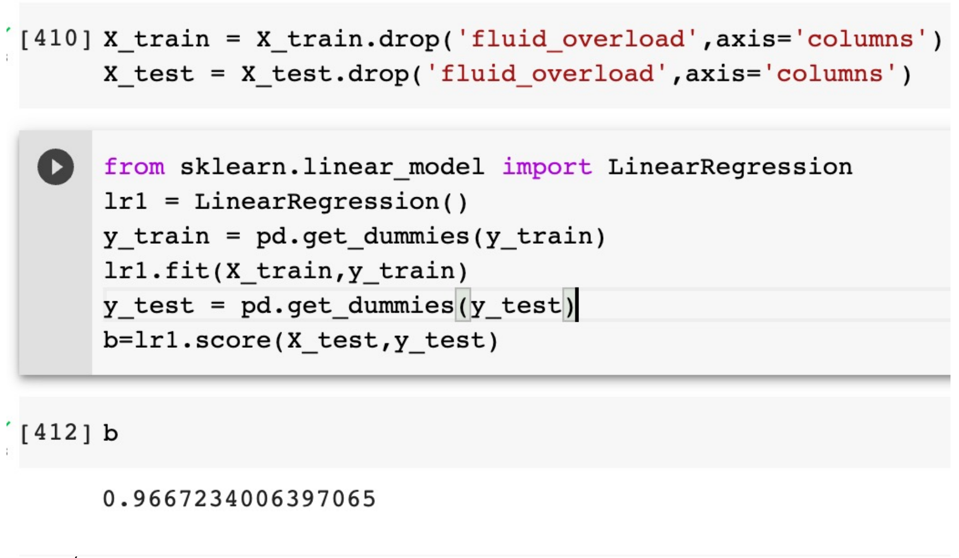
Visualizing The Score with and without “fluid_overload”
- Indicating no change to the accuracy



Linear Regression

```
[398] from sklearn.linear_model import LinearRegression
[399] lr = LinearRegression()
      y_train = pd.get_dummies(y_train)
[401] lr.fit(X_train,y_train)
      ▾ LinearRegression
      LinearRegression()
[402] y_test = pd.get_dummies(y_test)
[403] a=lr.score(X_test,y_test)
[404] a
      0.9667341432134509
```





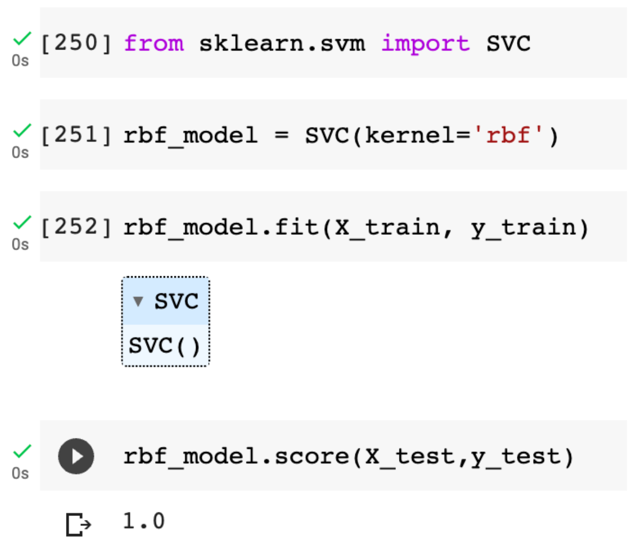
```
[410] X_train = X_train.drop('fluid_overload',axis='columns')
      X_test = X_test.drop('fluid_overload',axis='columns')

      from sklearn.linear_model import LinearRegression
      lr1 = LinearRegression()
      y_train = pd.get_dummies(y_train)
      lr1.fit(X_train,y_train)
      y_test = pd.get_dummies(y_test)
      b=lr1.score(X_test,y_test)

[412] b
      0.9667234006397065
```

SUPPORT VECTOR MACHINES (SVM)

- Support vector machines (SVMs) are a type of supervised learning algorithm used for classification and regression
- SVMs work by finding a hyperplane that best separates classes in the data
- Kernel functions are used to transform the data into a higher-dimensional space where it can be linearly separable
- To predict the accuracy of the Prognosis, we use the Training dataset and Prognosis as the target variable

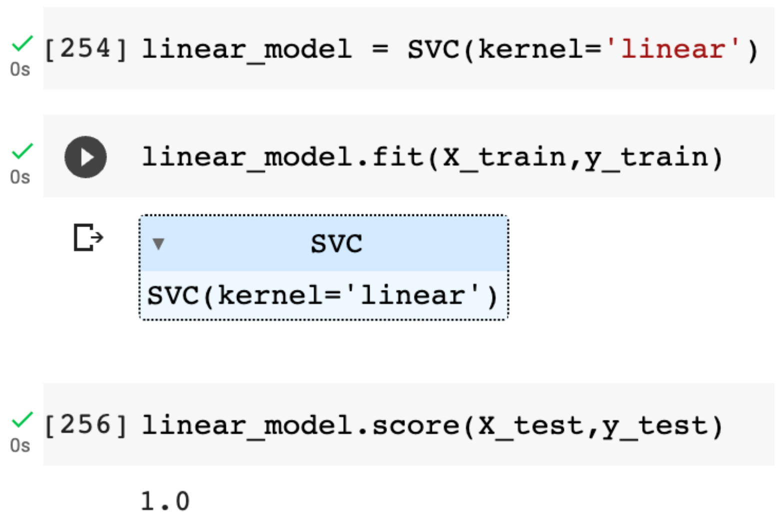
CODE SNIPPET – SVM

RBF



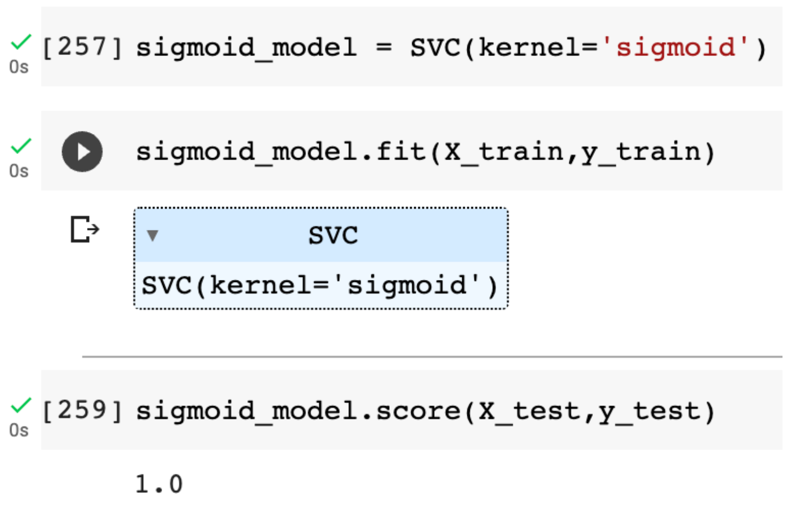
```
[250] from sklearn.svm import SVC

[251] rbf_model = SVC(kernel='rbf')

[252] rbf_model.fit(X_train, y_train)

      ▾ SVC
      SVC()

      rbf_model.score(X_test,y_test)

      1.0
```

Linear



```
[254] linear_model = SVC(kernel='linear')

      linear_model.fit(X_train,y_train)

      ▾ SVC
      SVC(kernel='linear')

[256] linear_model.score(X_test,y_test)

      1.0
```

Sigmoid

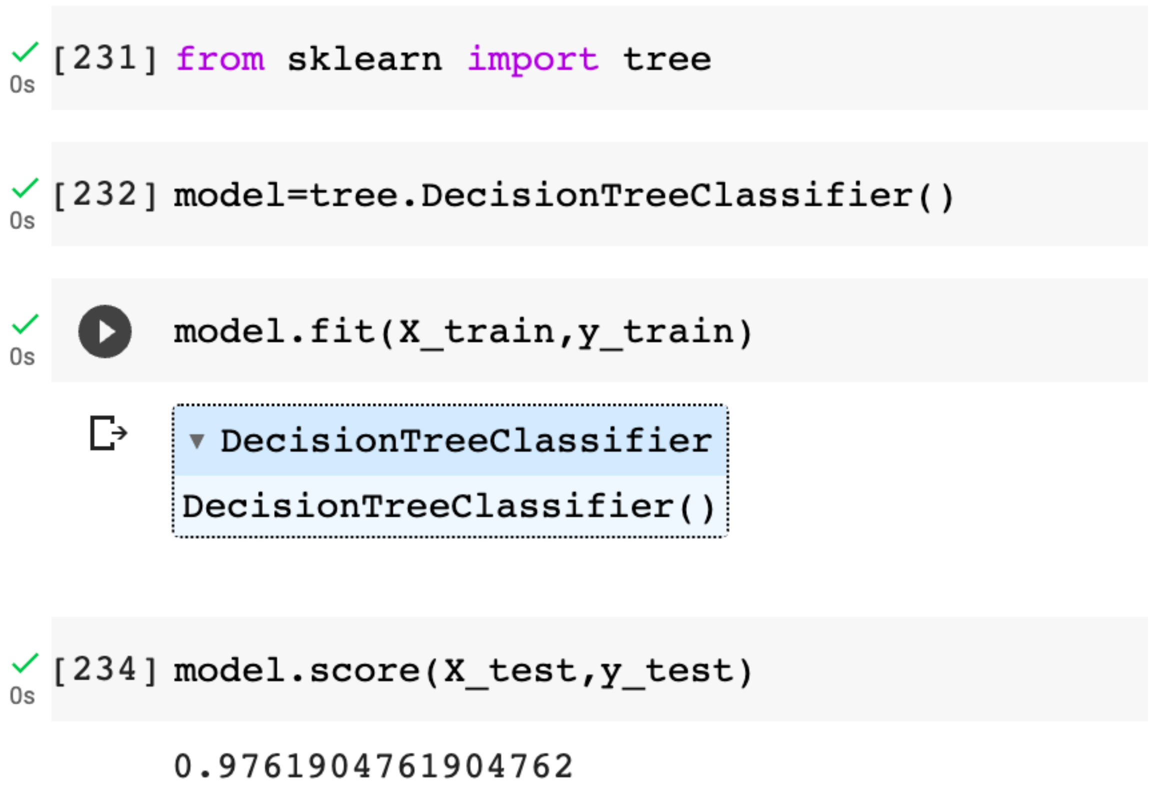


```
[257] sigmoid_model = SVC(kernel='sigmoid')

      sigmoid_model.fit(X_train,y_train)

      ▾ SVC
      SVC(kernel='sigmoid')

[259] sigmoid_model.score(X_test,y_test)

      1.0
```

DECISION TREE

- A Decision Tree is a machine learning algorithm used for both regression and classification tasks. It is a simple and intuitive model that builds a tree-like structure of decisions and their possible consequences.
- The basic idea behind Decision Trees is to recursively partition the data into subsets that are as homogeneous as possible with respect to the target variable.
- Use decision tree algorithm to predict how the independent variables affects the target variable
- To predict the accuracy of the Prognosis, we use the Training dataset and Prognosis as the target variable

CODE SNIPPET – DECISION TREE



```
[231] from sklearn import tree
```

```
[232] model=tree.DecisionTreeClassifier()
```

```
model.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

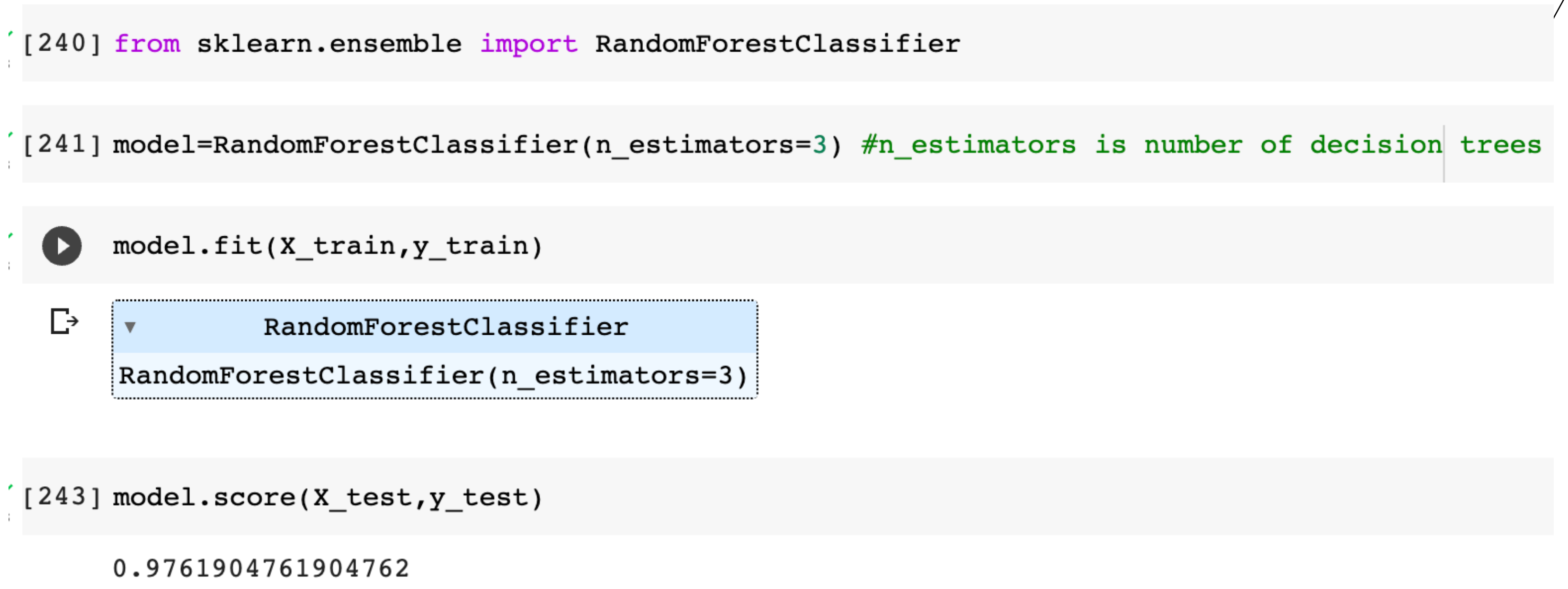
```
[234] model.score(X_test,y_test)
```

```
0.9761904761904762
```

RANDOM FOREST

- Random Forest is a machine learning algorithm used for both regression and classification tasks. It is an ensemble method that combines multiple decision trees to make a prediction.
- The basic idea behind Random Forest is to create a forest of decision trees, each of which is trained on a random subset of the training data and a random subset of the features. The final prediction is made by aggregating the predictions of all the individual trees.
- To predict the accuracy of the Prognosis, we use the Training dataset and Prognosis as the target variable

CODE SNIPPET – RANDOM FOREST



```
[240] from sklearn.ensemble import RandomForestClassifier
```

```
[241] model=RandomForestClassifier(n_estimators=3) #n_estimators is number of decision trees
```

```
model.fit(X_train,y_train)
```

```
▾       RandomForestClassifier
RandomForestClassifier(n_estimators=3)
```

```
[243] model.score(X_test,y_test)
```

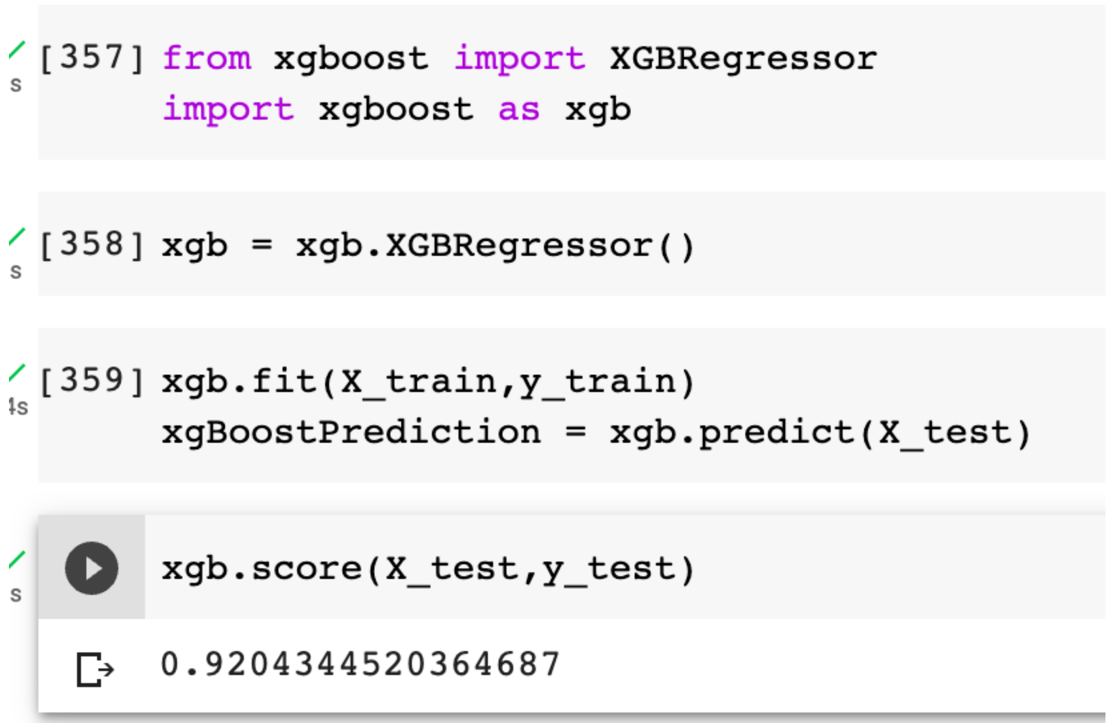
```
0.9761904761904762
```

XGBOOST

- XGBoost(Extreme Gradient Boosting), is an ensemble machine learning algorithm used for classification and regression tasks
- It uses a gradient boosting framework and employs multiple decision trees to make predictions
- XGBoost is known for its speed, accuracy, and ability to handle large datasets with high dimensionality

CODE SNIPPET - XGBOOST

Visualization



```
[357] from xgboost import XGBRegressor
      import xgboost as xgb
```

```
[358] xgb = xgb.XGBRegressor()
```

```
[359] xgb.fit(X_train,y_train)
      xgBoostPrediction = xgb.predict(X_test)
```

```
xgb.score(X_test,y_test)

0.9204344520364687
```

PERFORMANCE EVALUATION

Linear Regression : 0.966

SVM: 1

Decision Tree: 0.971

Random Forest: 0.976

XGBoost: 0.92

CONCLUSION

As per the comparative analysis of using these algorithms on the symptom-based classification problem, it is concluded that SVM fits the best as the accuracy score is 1

SVM outperformed Decision Tree in terms of classification accuracy, precision, and recall. SVM's ability to handle high-dimensional data and find the optimal hyperplane to separate the classes in the feature space made it a suitable choice for this problem.

Decision Tree had the advantage of being easier to interpret and visualize, but it lacked the accuracy and robustness of SVM

CONTRIBUTION

Soham Shah – 100%

Arundhati Pathrikar – 100%

Mamtha Shetty – 100 %

Shantanu Pingale – 100%

THANK YOU