Prediction model for Kidney disease

DS 597

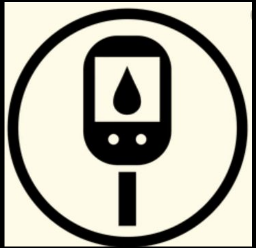
Sarwat Zabeen

Aditi Poddar

Introduction

- Kidney disease is a global public health problem and is related to serious mortality.
- In 2019, number of cases worldwide was 69.7 million.
- The global prevalence of Kidney disease was 9.1% in 2019.

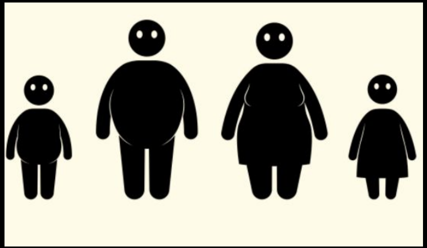
Diabetes



High blood pressure

Smoking

Obesity



BUSINESS PROBLEM

- Over 1.4 million patients receiving renal replacement therapy worldwide
- Lack of conclusive research on any early detection system based on lifestyle/behavioral factors
- Explore whether demographic and lifestyle factors can lead to prevention or early detection of kidney disease.

BUSINESS IMPACT

- Organizations in the healthcare space can benefit profoundly from this research and monitor early risk factors or risky behavioral patterns.

Approach and data Source

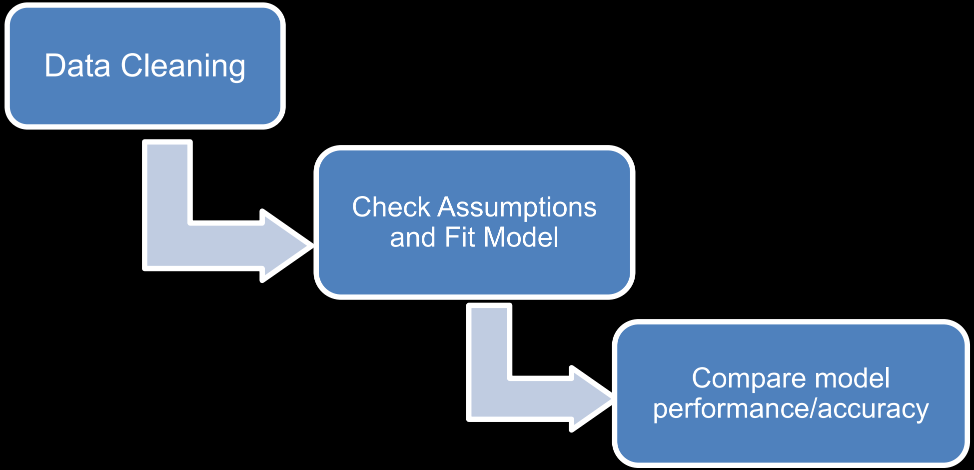
- Secondary Data
- The Behavioral Risk Factor Surveillance System data is a dataset available directly from the CDC website
- Published Yearly

Tools and Techniques

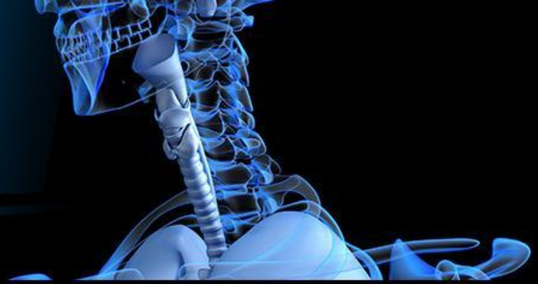
- Five predictive models utilized. These are
- logistic regression
- random forest
- decision tree
- SVM
- naive Bayes

Methodology

- Python utilized for data cleaning and only the variables relevant to this analysis will be retained



Expected Results



- From this project we intend to design a model which will forecast the early detection of chronic kidney disease.
- Determination of factors linked with kidney disease.
- Using Decision tree model, data set will be broken down into smaller subsets to identify useful data.

TIMELINE SLIDE

Week1

Literature review
Planning the Step/ procedure

Week2

Data Cleaning

Week3

Using Proper Sampling Technique
Exploratory Data Analysis

Week4

Checking Model Assumption

Week5

Run Model and Compare results

TIMELINE SLIDE

Hyperparameter Turning
ROC AUC

Compare Results Again

Week6

Week7

Week8

Week9

Week10

Run Models

Try Different Sampling technique

Results and Future Directions

Data cleaning

- We replaced missing values with mean value
- Data imbalanced

Dealing with Imbalanced Dataset

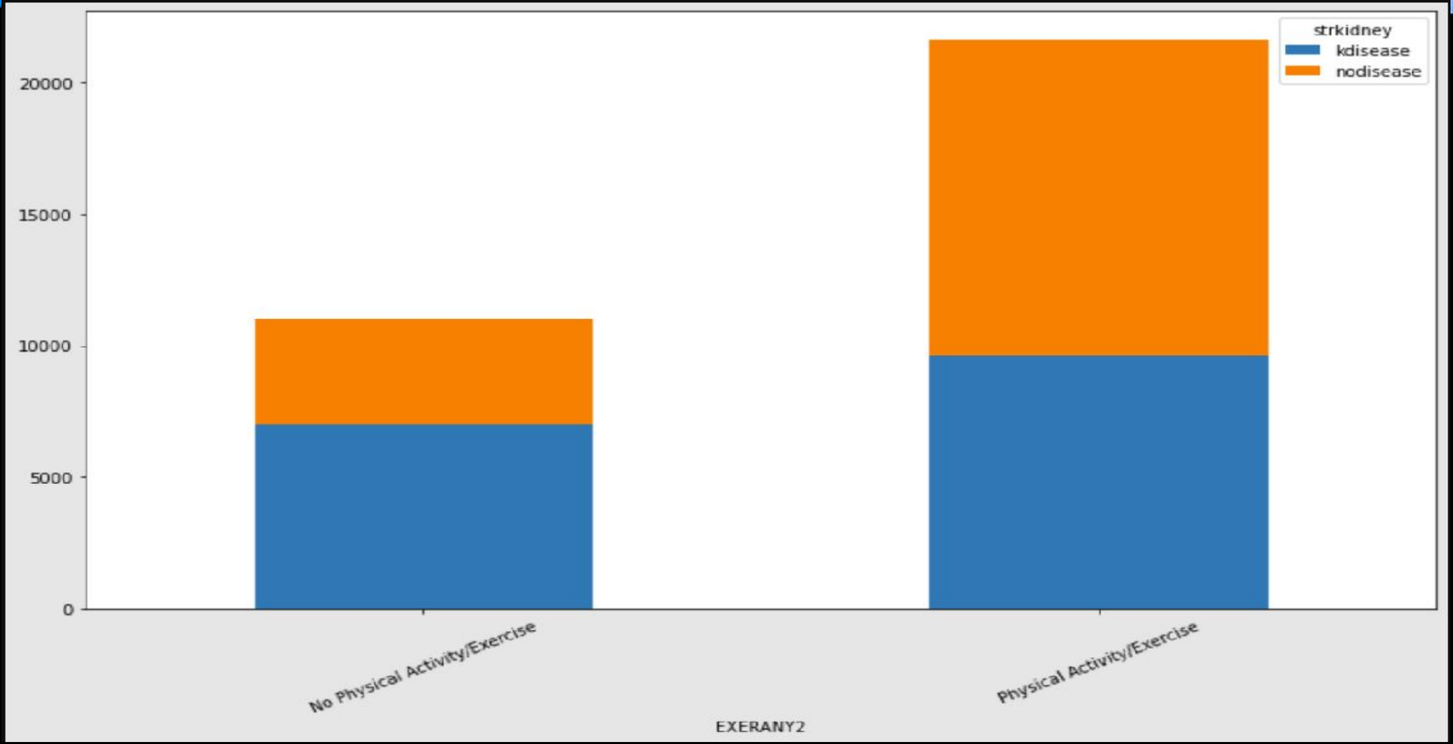
Data cleaning contd.

- Data imbalanced
- 13,332 participants have kidney diseases and 340,269 participants do not have kidney diseases.
- Respondents who have kidney diseases are approximately four percent of the total.

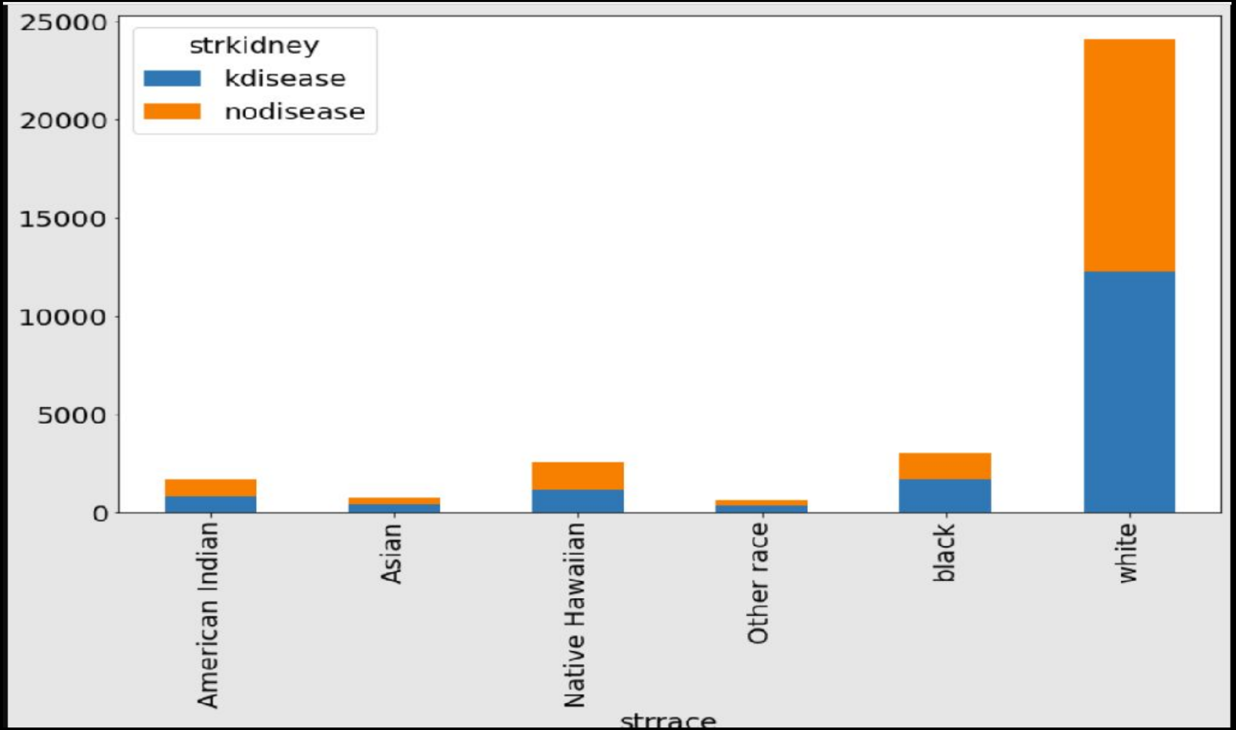
Kidney Disease by State

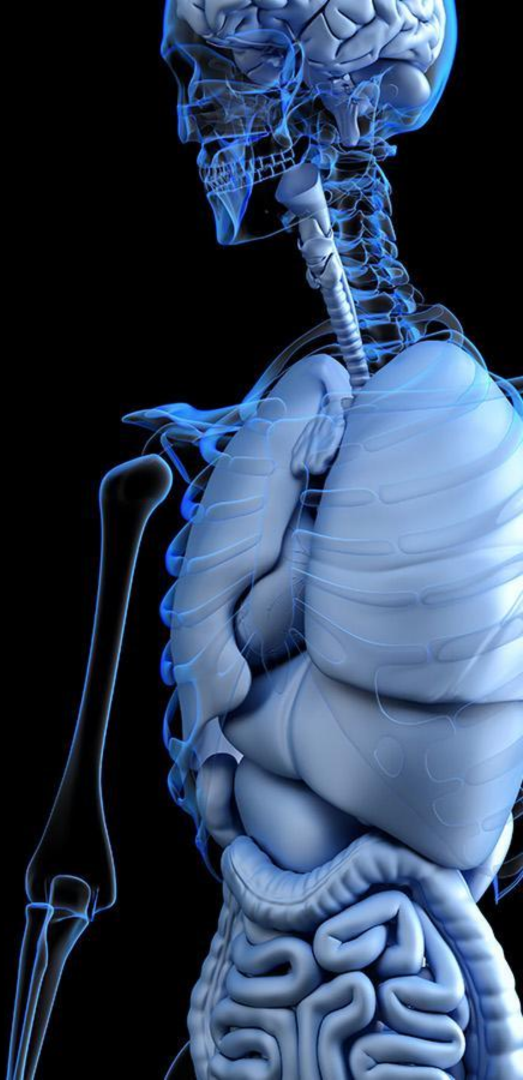
[Kidney Disease Prediction - Sarwat Zabeen | Tableau Public]()

Kidney Disease Vs Exercise

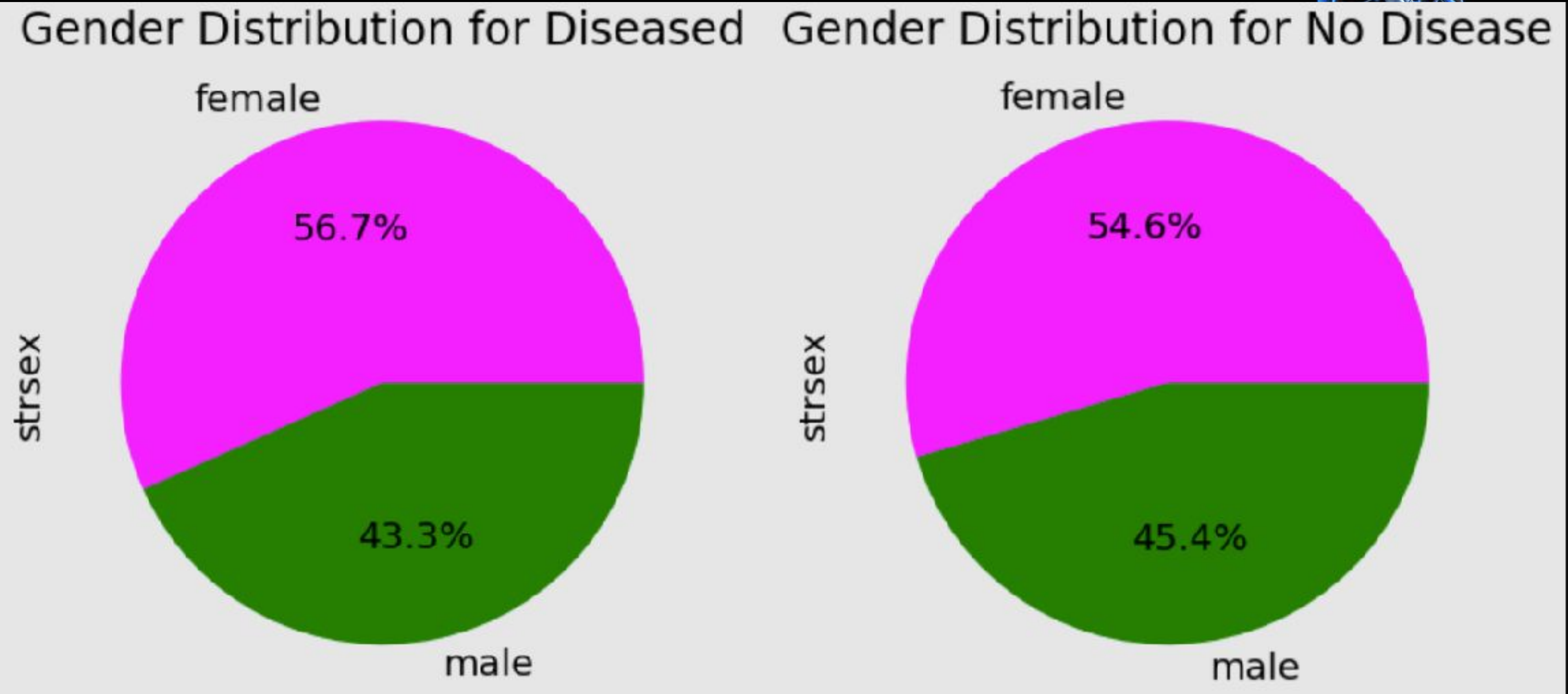


Race and Kidney disease

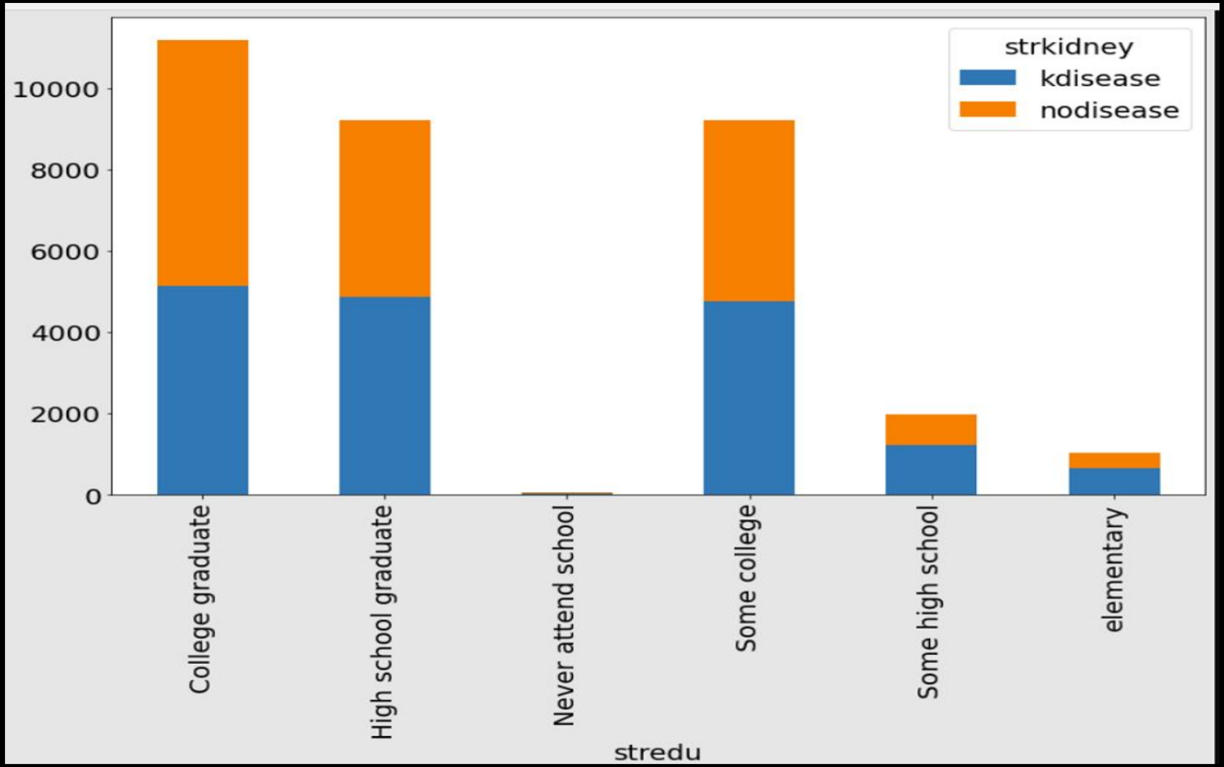


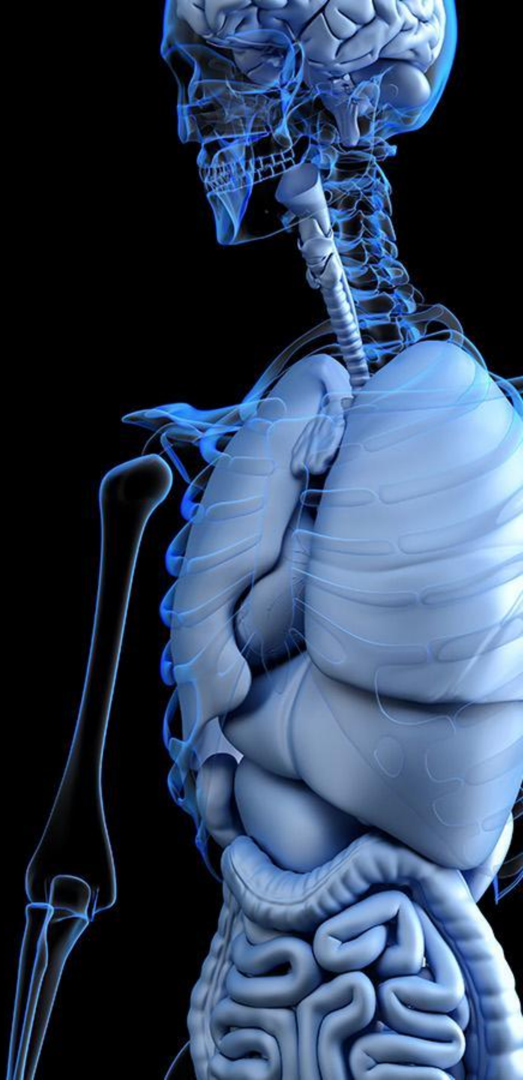


Sex and Kidney disease

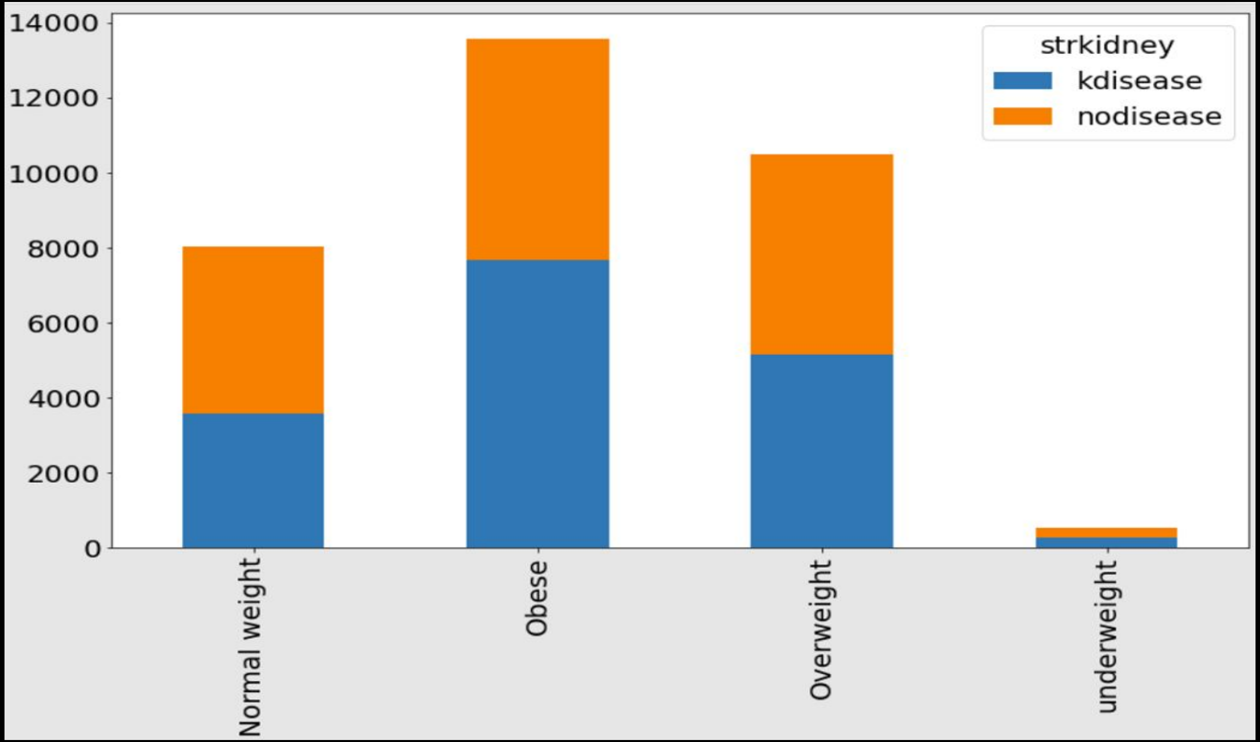


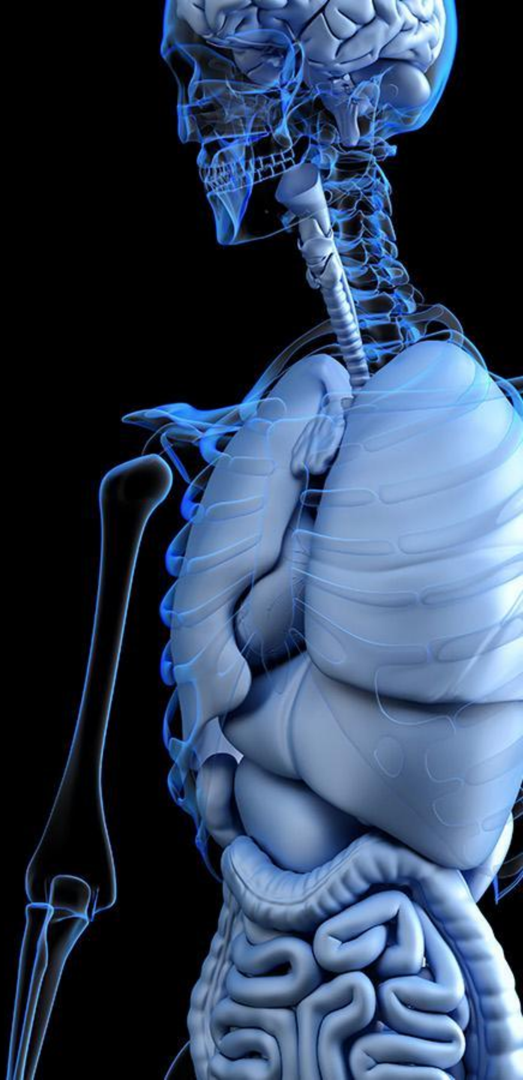
Education and Kidney disease





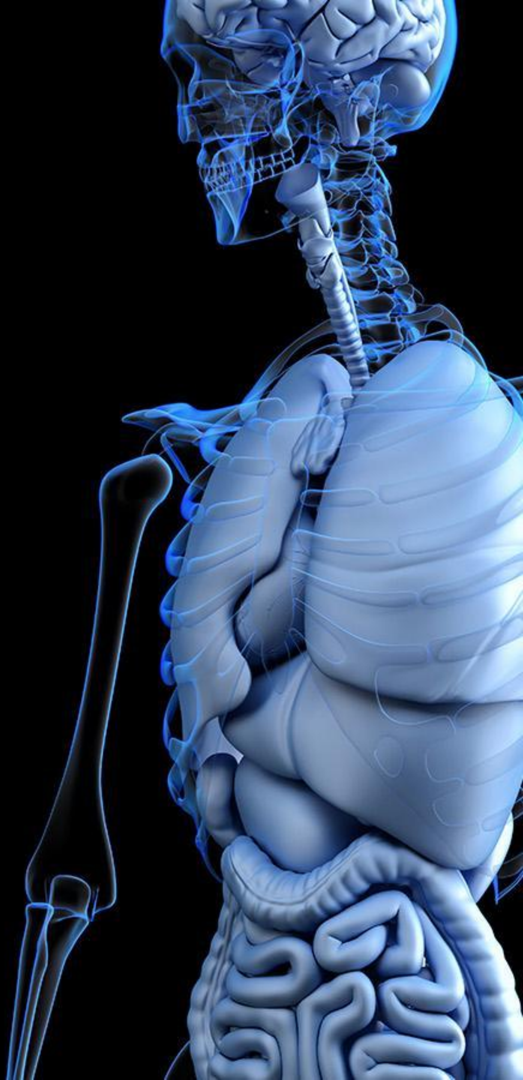
BMI and Kidney Disease



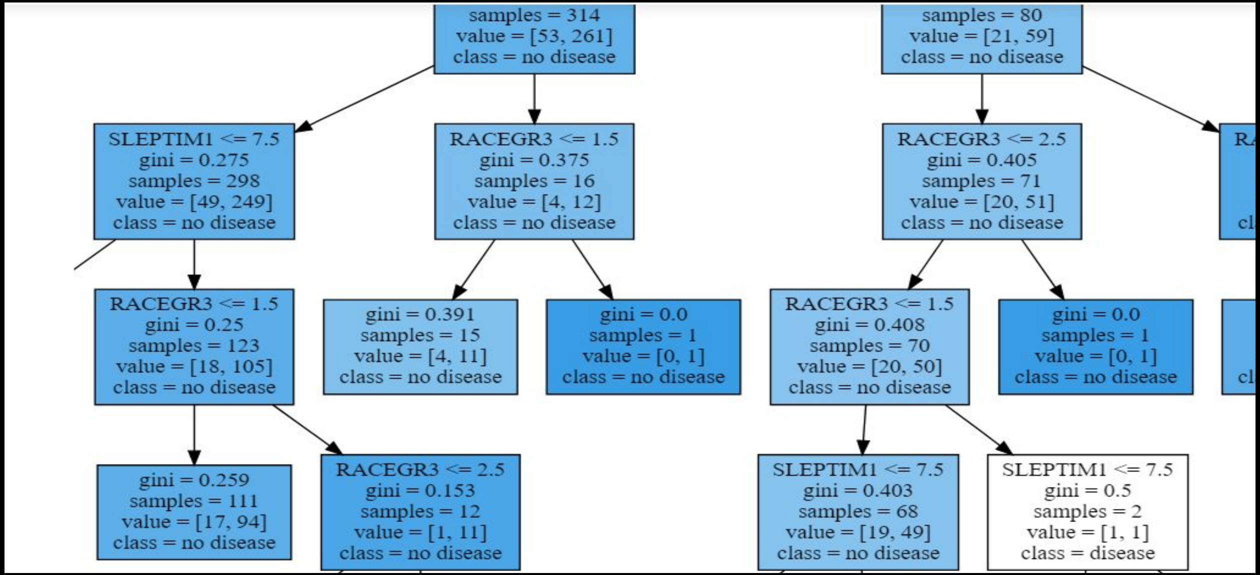


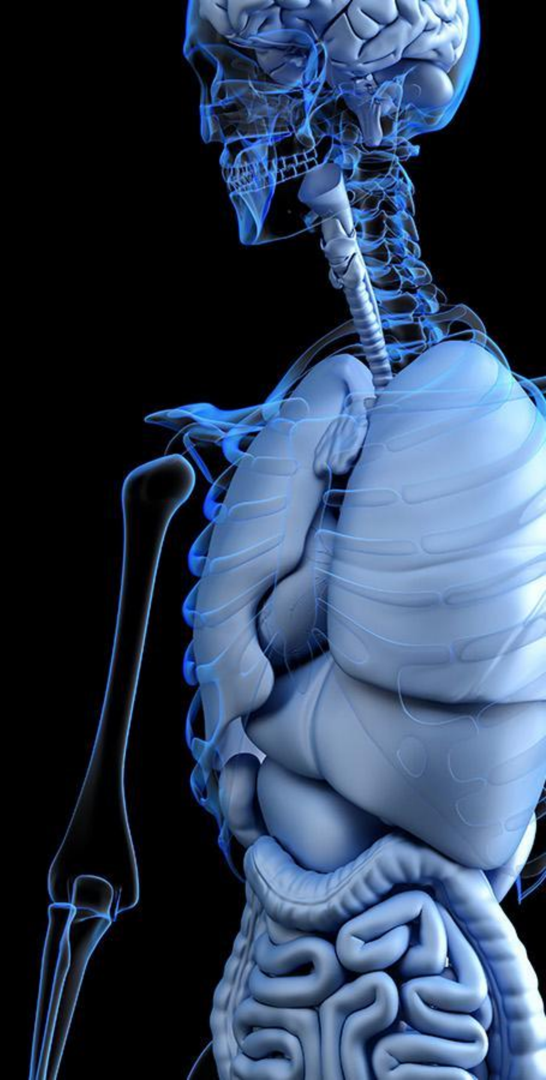
Logistic Regression Full Model

```
In [117]: #Split dataset into training and test object
          x_train, x_test, y_train, y_test=train_test_split(X, Y, random_state=1)

In [118]: x_train.shape

Out[118]: (24485, 12)

In [119]: #Create a Logistic Regression Object, perform Logistic Regression

          log_reg=LogisticRegression(max_iter=1200000)
          log_reg.fit(x_train, y_train)

Out[119]: LogisticRegression(max_iter=1200000)

In [120]: y_pred=log_reg.predict(x_test)

In [124]: print(confusion_matrix(y_test, y_pred))

          print(accuracy_score(y_test, y_pred)*100)

          [[3012 1044]
           [1628 2478]]
          67.26292575349179
```



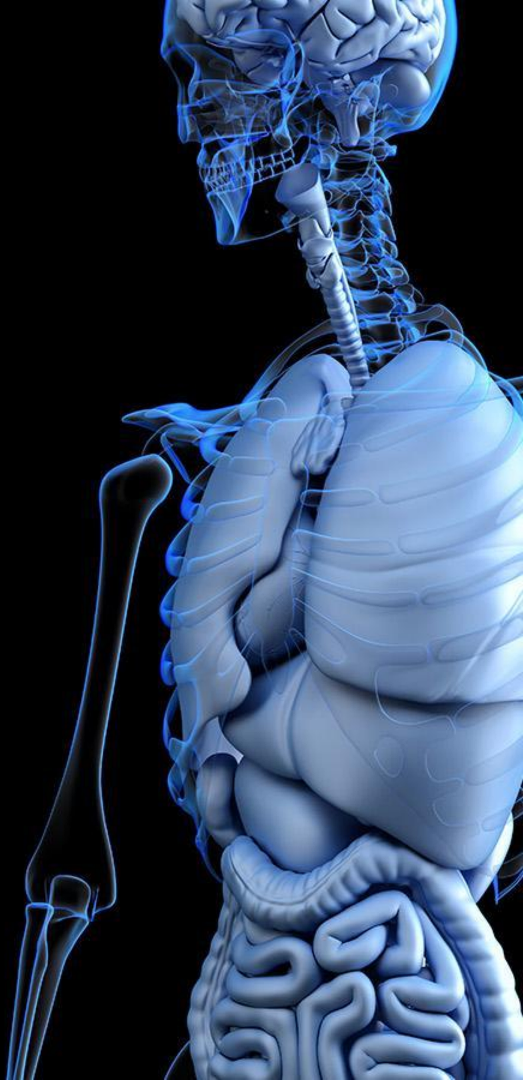
Visualizing the tree(How Race is branching out)





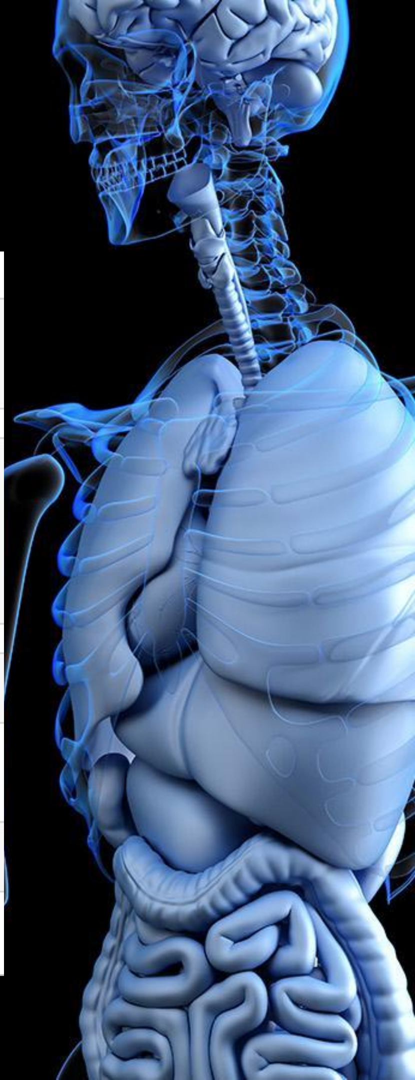
RF model confusion matrix

	Disease Present	Disease Absent
Detected by model	2177	1094
Undetected by model	1283	1976

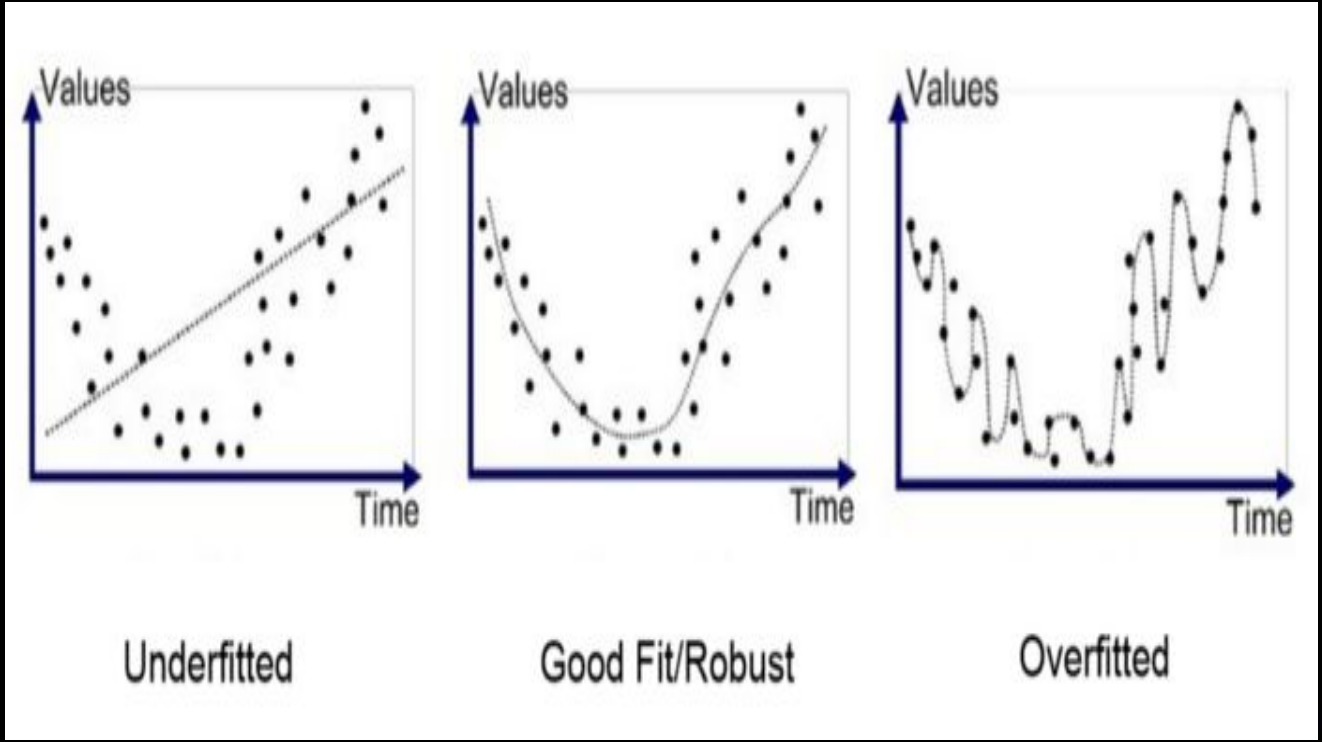


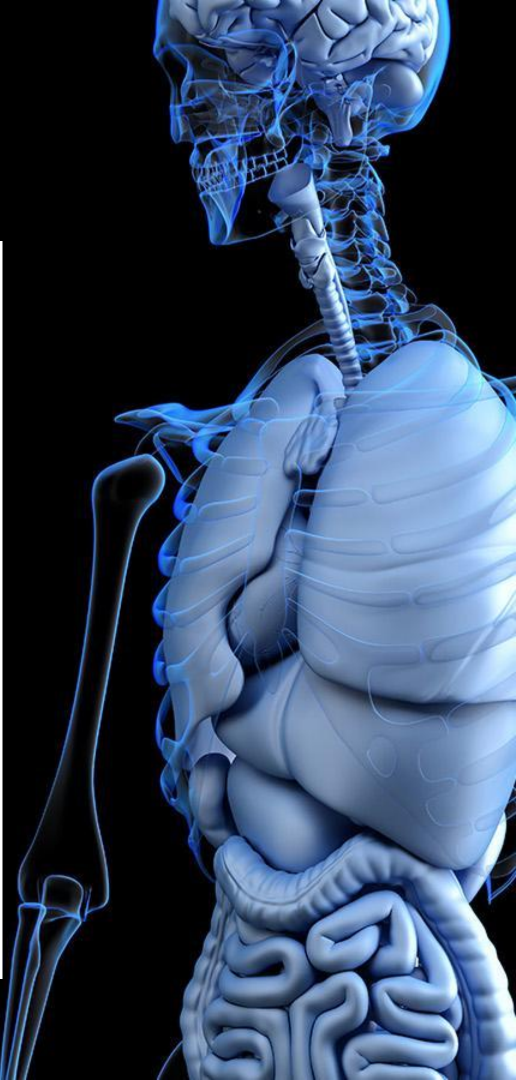
SVM with C=100

```
In [51]: from sklearn.svm import SVC
         model = SVC(C=100)

In [52]: X = df_nostring.drop('CHCKDNY1', axis=1)
         Y = df_nostring['CHCKDNY1']

         x_train, x_test, y_train, y_test=train_test_split(X, Y, test_size=0.2)

In [53]: model.fit(x_train, y_train)

Out[53]: SVC(C=100)

In [54]: model.score(x_test, y_test)

Out[54]: 0.650229709035222
```



Regularization and Overfitting





Try different hyperparameter

In [68]:
```python
#Try different hyperparameters

#no. of grids in rf
n_estimators = [int(x) for x in np.linspace(start=10, stop=100, num=10)]

criterion = ['gini', 'entropy']

#no. of features to consider at every split
max_features=['auto', 'sqrt']

#maximum number of levels in tree

max_depth = [3,5,7,9,10]

#minimum number of samples required to split a node

min_samples_split = [2,4,6]

#minimum number of samples required at each leaf node

min_samples_leaf = [1, 2, 4]

#method of selecting samples for training each tree

bootstrap = [True, False]
```

In [69]:
```python
#create the new random grid
```

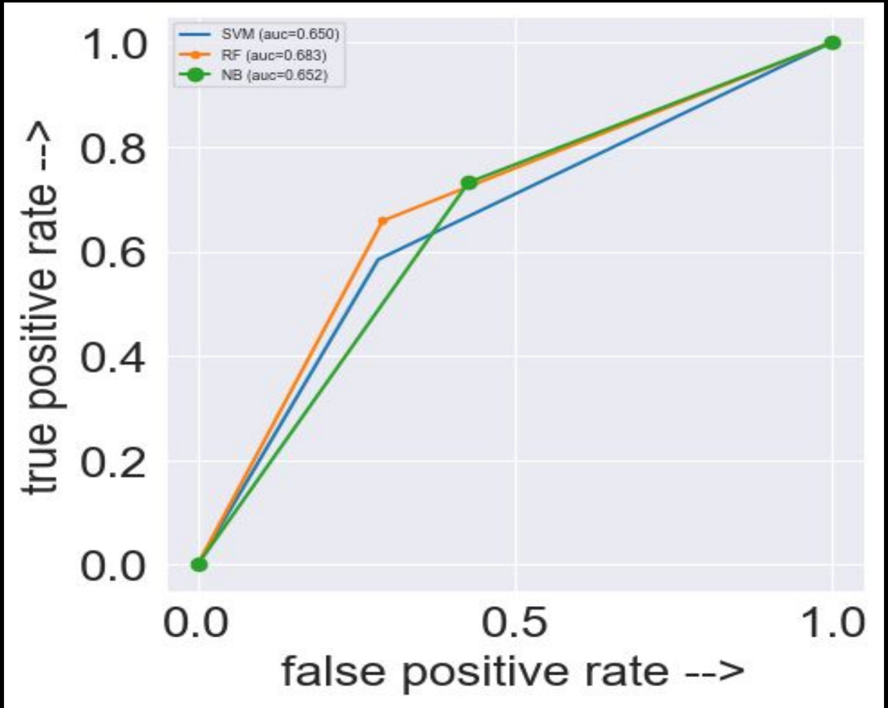
Best parameters

```
              verbose=2)

In [79]: rf_grid.best_params_

Out[79]: {'bootstrap': True,
          'criterion': 'entropy',
          'max_depth': 7,
          'max_features': 'sqrt',
          'min_samples_leaf': 2,
          'min_samples_split': 2,
          'n_estimators': 70}

In [80]: rf_grid.score(x_train, y_train)
         rf_grid.score(x_test, y_test)

Out[80]: 0.683920367534456
```

NB output

```
In [81]: from sklearn.naive_bayes import GaussianNB

In [82]: nb = GaussianNB()
         nb.fit(x_train, y_train)

Out[82]: GaussianNB()

In [83]: nb.score(x_train, y_train)
         nb.score(x_test, y_test)

Out[83]: 0.6506891271056662
```

ROC AUC

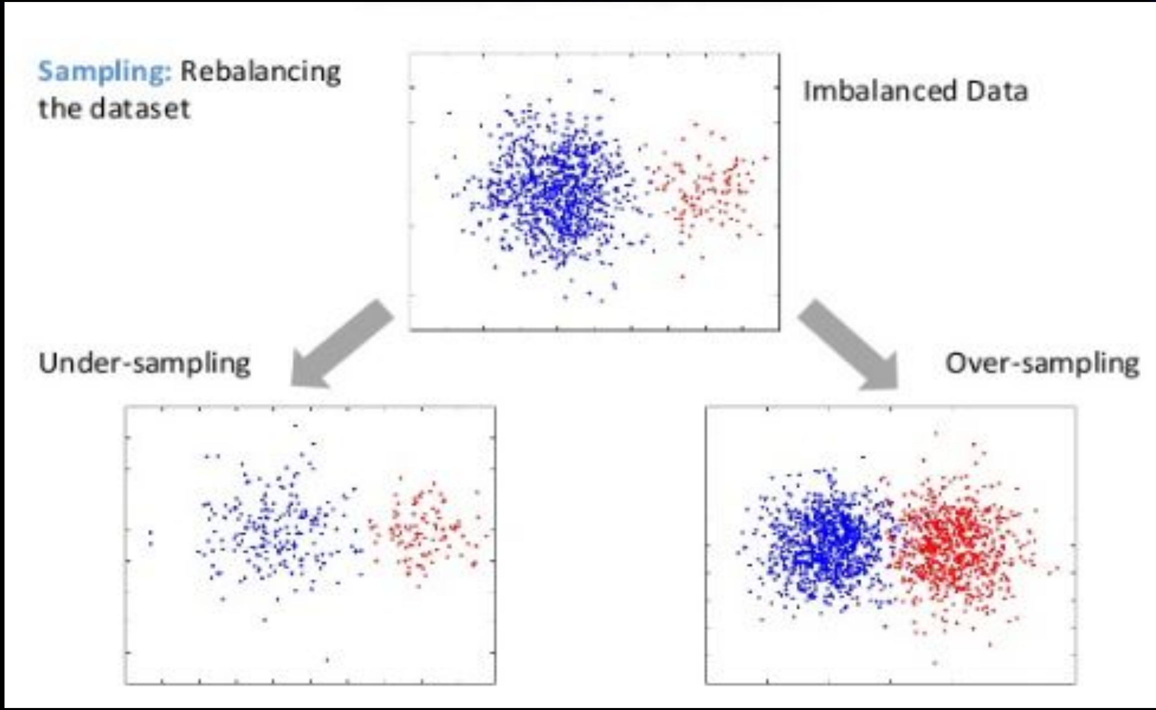


Rf_auc	75.5%
Nb_auc	71.4%
SVM_auc	71.1%

Let’s compare Again

	LR Full Model	LR Reduced Model	Decision Tree	Random Forest	SVM	NB
Accuracy	67%	58%	64%	68.4%	65%	65%
False Positives	1044	2298	1494	1094	947	1364
False Negatives	1628	1077	2015	1283	1379	895

Oversampling Vs Undersampling



Advantages and disadvantages

Advantages:

- Does not discard potentially useful data
- Rich, more representative of the population

Disadvantages:

- Overfitting likely
- Increases learning time

Decision Tree

```
In [37]: from sklearn import tree
         from sklearn.tree import DecisionTreeClassifier

         from sklearn.tree import export_graphviz
```

```
In [44]: x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=15, stratify=Y

         model = tree.DecisionTreeClassifier()
         model = model.fit(x_train, y_train)
```

```
In [45]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score

         y_predict=model.predict(x_test)

         print(confusion_matrix(y_test, y_predict))
         print(accuracy_score(y_test, y_predict)*100)

         [[125146    605]
          [  9382 116368]]
         96.02904163402928
```

Results/Conclusion

- Likelihood of getting kidney disease can be predicted using factors such as sex, race, BMI, exercise, smoking, education, amount of sleep and employment status.
- Decision Tree is the most accurate model in making such predictions. However, should be used with caution since RF is lower

Direction for Future Work

- National Kidney Foundation highlights the important of drinking adequate water in preventing kidney disease
- It also cites too many OTC painkillers as a common cause of kidney disease
- CDC can incorporate these measures into BRFSS

Thank you