REPORT OF LUNG CANCER DETECTION MODEL EVALUATION

Project Topic

This project is about creating lung cancer detection using Supervised Machine learning Algorithms. This project outcome is to help the physicians to help and save time in identifying the nodules present in CT lung images in the early stage of lung cancer. Since that process is the time consuming and very critical

Motive

The effectiveness of the cancer prediction system helps people to know their cancer risk at a low cost and it also helps the people to take the appropriate decision based on their cancer risk status. The data is collected from the website online lung cancer prediction system.

Data

Data is a Public kaggle dataset which is called Lung Cancer.
source link:
https://www.kaggle.com/datasets/nancyalaswad90/lung-cancer/data

In-Text Citation: (Hong & Yang, 1991)

Reference List Entry:

Hong, Z. Q., & Yang, J. Y. (1991). Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane Lung Cancer Dataset. Kaggle.
https://www.kaggle.com/datasets/nancyalaswad90/lung-cancer/data

License: CC BY-NC-SA 4.0

Format: CSV

Size:

RangeIndex: 309 entries,
Data Types: int64(14), object(2)

```
data.info()
```

Data Cleaning

```
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   GENDER                 309 non-null    object
 1   AGE                    309 non-null    int64 
 2   SMOKING                309 non-null    int64 
 3   YELLOW_FINGERS         309 non-null    int64 
 4   ANXIETY                309 non-null    int64 
 5   PEER_PRESSURE          309 non-null    int64 
 6   CHRONIC DISEASE        309 non-null    int64 
 7   FATIGUE                309 non-null    int64 
 8   ALLERGY                309 non-null    int64 
 9   WHEEZING               309 non-null    int64 
 10  ALCOHOL CONSUMING      309 non-null    int64 
 11  COUGHING               309 non-null    int64 
 12  SHORTNESS OF BREATH    309 non-null    int64 
 13  SWALLOWING DIFFICULTY  309 non-null    int64 
 14  CHEST PAIN             309 non-null    int64 
 15  LUNG_CANCER            309 non-null    object
```

Data doesn’t contain any null values in all of the columns and only two columns are object types while all of them are 64 bits integer type.We compute this by using the info method on dataframe object.

Summary

Age Distribution: The dataset covers a range of ages from 21 to 87, with the majority (50%) failing between 57 and 69 years.

Categorical Variables: The statistics for other variables suggest that they might represent binary attributes.

For the detail of data distribution please check the output of

dataset.describe() row.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical initial step in the data analysis process that involves exploring and summarizing the main characteristics, patterns, and relationships present in a dataset. It's performed to gain insights, discover patterns, and identify potential issues or trends within the data.

Why Perform EDA?

EDA helps understand the structure, nature, and content of the dataset. It gives an overview of what the data contains and how it's organized.

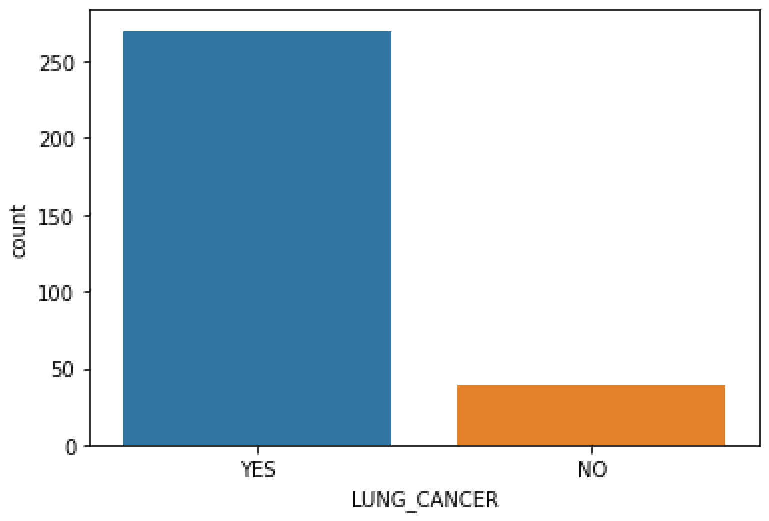
EDA uncovers patterns, trends, correlations, and relationships among variables, helping to generate hypotheses and insights.

It helps identify potential errors, outliers, missing values, or inconsistencies in the data that may impact subsequent analyses or modeling.

EDA aids in selecting relevant features (variables) for modeling and in creating new informative features through feature engineering.

Analyzing Lung Cancer Column

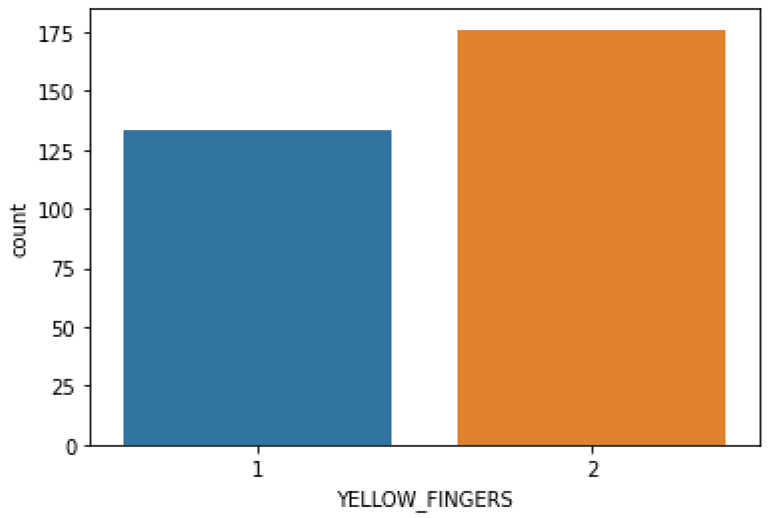
```
sns.countplot(x = 'LUNG_CANCER',data = dataset)
```



The data has more present lung cancer data. So the predicted value can be biased to the more data we need to resampled for this. I will show the resampled in the following of the report after splitting the data into train and test sets.

Analyzing YELLOW FINGERS Column

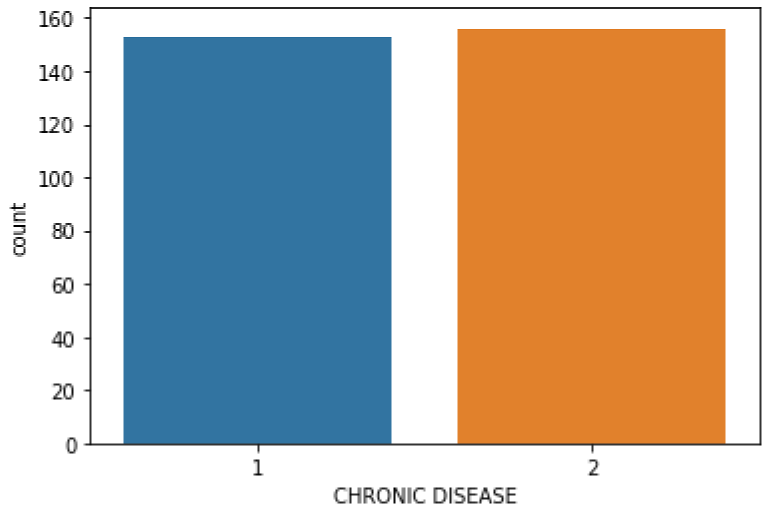
```
sns.countplot(x = 'YELLOW_FINGERS',data = dataset)
```



In this column the data is a little bit unbalanced.

Analyzing CHRONIC DISEASE Column

```
sns.countplot(x = 'CHRONIC DISEASE',data = dataset)
```



The data are nearly equally distributed.

Label Encoding

```
le = preprocessing.LabelEncoder()
dataset['GENDER'] = le.fit_transform(dataset['GENDER'])
dataset['LUNG_CANCER'] = le.fit_transform(dataset['LUNG_CANCER'])
```

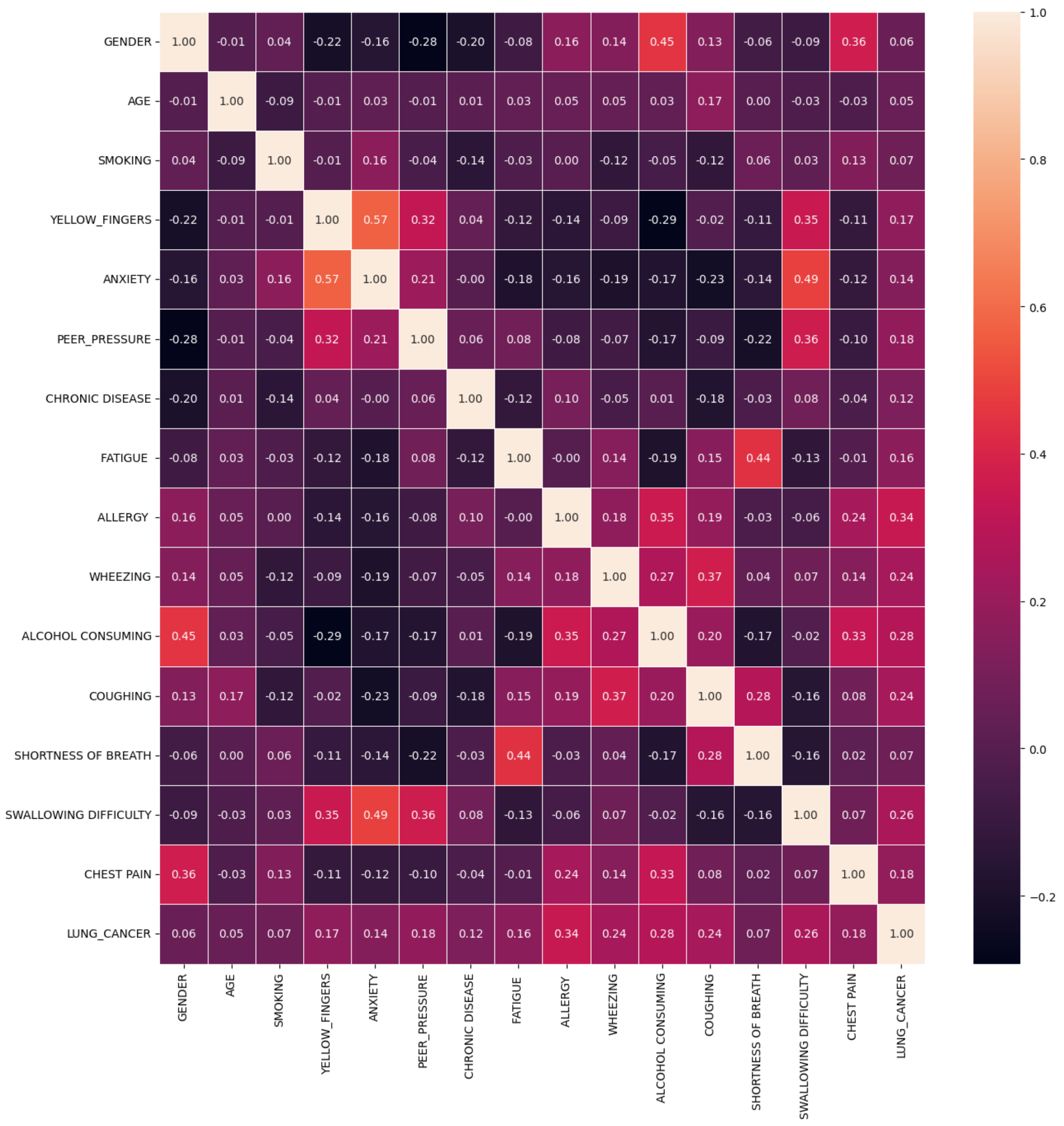
Label Encoding is needed for two objects such as LUNG_CANCER whose values are (Yes, NO) and GENDER (M,F). It will change the values from categories to numerical here (0,1). Label encoding is required for training the data. Now all the values are integers type.

Correlation Matrix

Here Anxiety and Yellow fingers (0.57) columns are more correlated than other columns and rest are (Yellow Fingers & Peer Pressure, 0.32

), (Anxiety & Swallowing Difficulties, 0.49), (Wheezing & coughing , 0.37), (Fatigue & Shortness of Breath, 0.44)

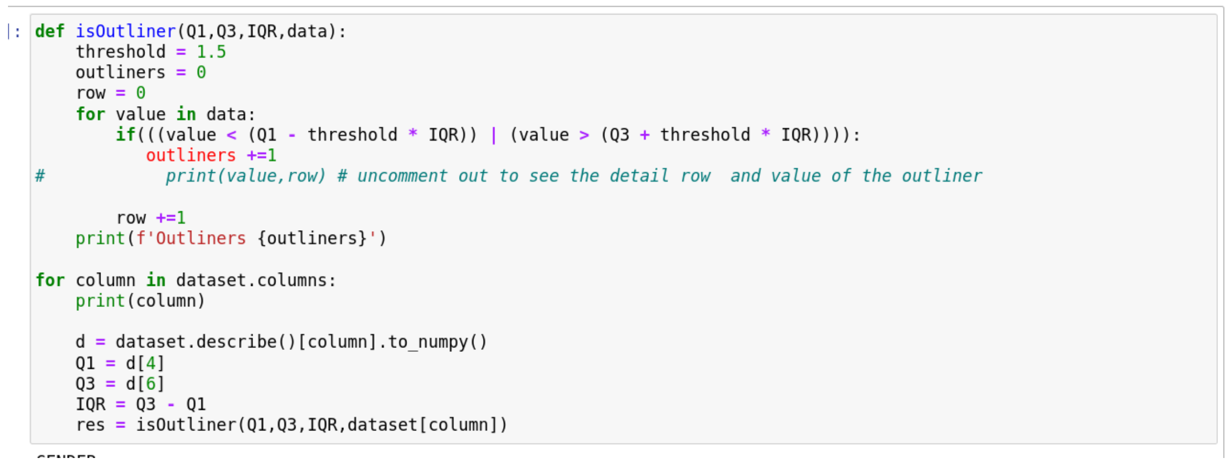
```
plt.figure(figsize=(18, 18))  # Set the size of the plot

# Customize the heatmap using Seaborn
sns.heatmap(correlation_matrix,annot=True, square=True, vmin=0, vmax=1,cmap="YlGnBu");
plt.title('Correlation Matrix')  # Set the title of the plot
plt.show()
```



There are multicollinearity present in the data such as ANXIETY & SWALLOWING DIFFICULTY, ANXIETY & YELLOW_FINGERS, PEER_PERSSURE & YELLOW_FINGERS.

It’s not possible to display the visualization of AGE vs Categorical columns in this report so please checkout the jupyter notebook for that.

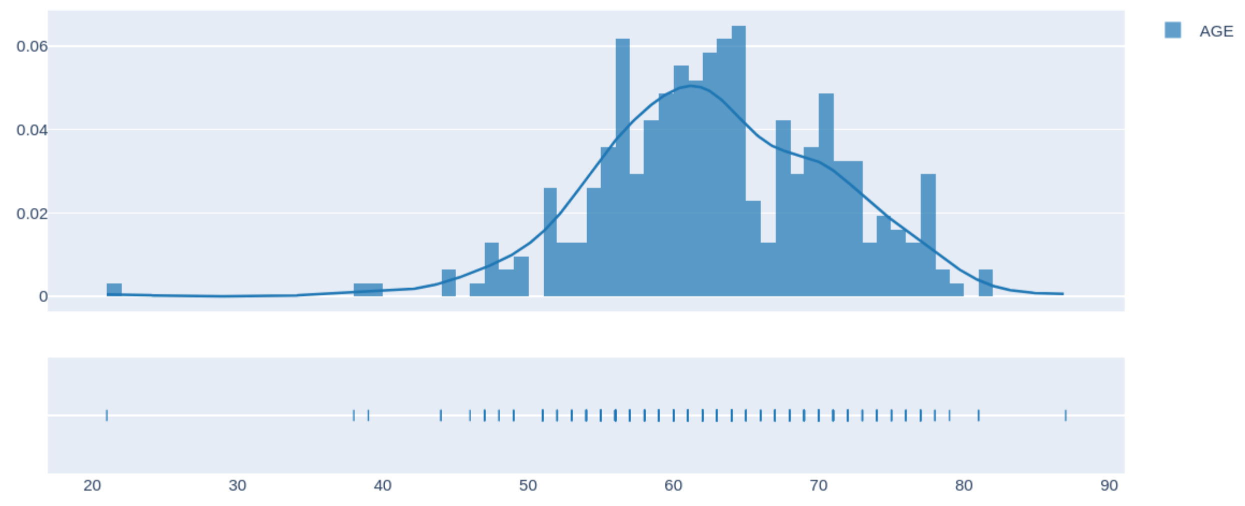
Detecting Outliers



```
def isOutliner(Q1,Q3,IQR,data):
    threshold = 1.5
    outliners = 0
    row = 0
    for value in data:
        if(((value < (Q1 - threshold * IQR)) | (value > (Q3 + threshold * IQR)))):
           outliners +=1
#            print(value,row) # uncomment out to see the detail row  and value of the outliner

        row +=1
    print(f'Outliners {outliners}')

for column in dataset.columns:
    print(column)

    d = dataset.describe()[column].to_numpy()
    Q1 = d[4]
    Q3 = d[6]
    IQR = Q3 - Q1
    res = isOutliner(Q1,Q3,IQR,dataset[column])
```

I used Interquartile range, Q1 AND Q3 to detect outliers in the dataset and found 2 in AGE column the value and row are like this

Value	Row
21	22
38	238



Histogram of AGE Column

Data Preprocessing

Let’s remove the outliers from the AGE column.

```
dataset = dataset.drop([22,238])
```

Prepare the data for training.

```
X=dataset.drop(['LUNG_CANCER'],axis=1)
y=dataset['LUNG_CANCER']
```

Changing the value of Category Variable from 2 and 1 to 1 and 0.

```
for i in X.columns[2:]:
   temp=[]
   for j in X[i]:
       temp.append(j-1)
   X[i]=temp
X.head()
```

OverSampling

Here, oversampling technique is used to rebalance the imbalance data in the dataset.

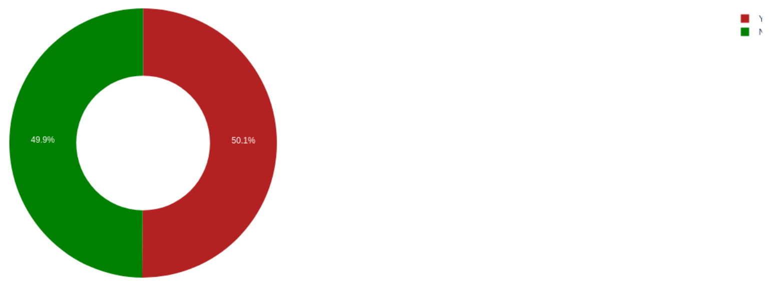
```
from imblearn.over_sampling import RandomOverSampler
X_over,y_over=RandomOverSampler().fit_resample(X,y)
```

Train & Test Splitting

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =
train_test_split(X_over,y_over,random_state=42,stratify=y_over)
print(f'Train shape : {X_train.shape}\nTest shape: {X_test.shape}')
```

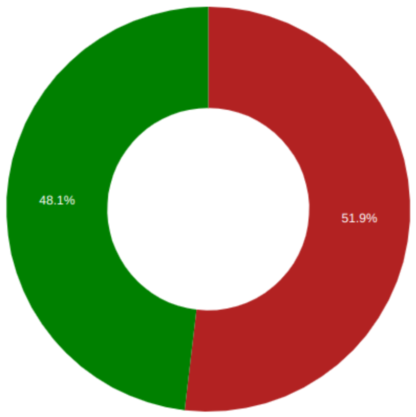
Data Distribution after oversampling.

```
values = y_train.value_counts().tolist()
names = ['Yes','No']

px.pie(y_train, values=values, names=names, hole = 0.5,
      color_discrete_sequence=["firebrick", "green"])
```



Now, the target column LUNG_CANCER is properly balanced after sampling.

```
values = X_train['YELLOW_FINGERS'].value_counts().tolist()
names = ['1','2']

px.pie(X_train['YELLOW_FINGERS'], values=values, names=names, hole = 0.5,
      color_discrete_sequence=["firebrick", "green"])
```



Above diagram is the data distribution of the Yellow Finger column after resampling on the Train Set which was significantly imbalanced on the original data.

MODEL & Result & Analysis

The model we need here is to predict binary classification (yes, no) for lung cancer. So we can't use the continuous outcome models and multiple regression models.

The models that I used in this project are

1. Random Forest Classifier
2. Decision Tree
3. Voting Classifier
4. Logistic Regression
5. Artificial Neural Network (ANN)
6. K Nearest Neighbour (KNN)

Here there is a new model which didn't cover in the class, the Voting Classifier (VC).

A Voting Classifier is an ensemble machine learning technique that combines the predictions from multiple individual machine learning models to make a final prediction. It aggregates the predictions of

each classifier and outputs the class label that received the most votes (or highest probability in the case of soft voting).
There are two main types of Voting Classifiers:

Hard Voting: In hard voting, the final prediction is determined by a simple majority vote among the individual classifiers. The class that receives the most votes from the individual classifiers is selected as the final prediction.

Soft Voting: In soft voting, each classifier's prediction is associated with a probability score for each class. The final prediction is determined by averaging the predicted probabilities for each class across all classifiers and selecting the class with the highest average probability.

Classifier Diversity: Voting classifiers work well when the individual classifiers are diverse in terms of their learning algorithms, data representation, or training process. When diverse classifiers are combined, they can compensate for each other's weaknesses and improve overall performance.

Performance Improvement: Voting classifiers can sometimes outperform individual classifiers by reducing variance and improving generalization. They tend to perform better when the individual classifiers have similar accuracy but make different types of errors.

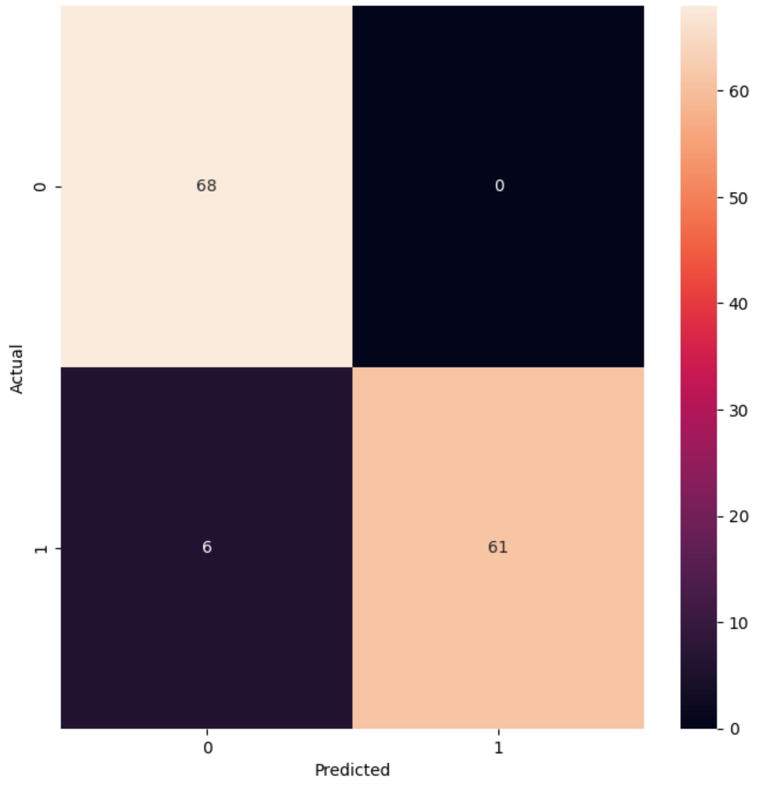
Usage: They are commonly used with a variety of classifiers, including decision trees, logistic regression, support vector machines (SVM), k-nearest neighbors (KNN), etc.
Implementation: In Python, sci

Here is the detailed implementation of the VC model.

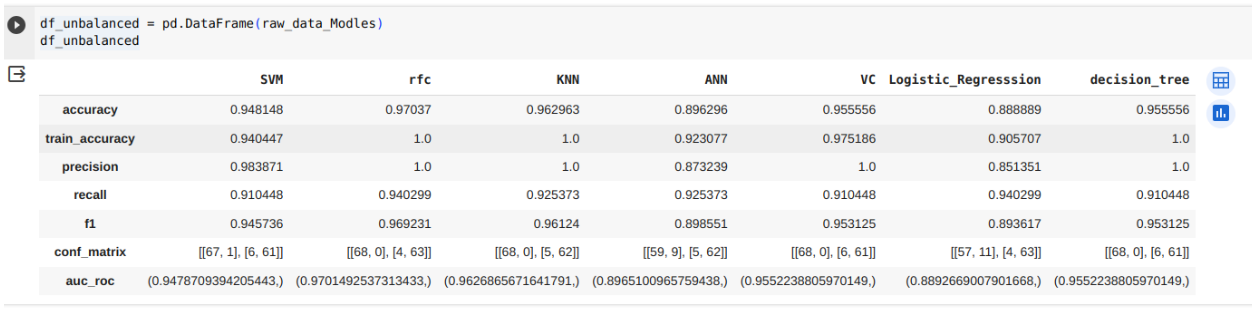
```python
def VC(X,y):
   clf1 = SVC()
   clf2 = KNeighborsClassifier()
   clf3 = RandomForestClassifier(n_estimators=100, random_state=42)

   eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb',
clf3)], voting='hard')
   eclf.fit(X,y)
```

```
    return eclf
```

```
vc  = VC(X_train,y_train)

predTrain,pred =  getPredPredTrain(vc)
metrics = getMetrics(y_test, pred,predTrain)
raw_data_Modles['VC'] = metrics
```



The accuracy, train_accuracy, precision, recall, f1 score, conf_matrix and auc_roc values for trained models are as follows.



```
df_unbalanced = pd.DataFrame(raw_data_Modles)
df_unbalanced
```

	SVM	rfc	KNN	ANN	VC	Logistic_Regresssion	decision_tree
accuracy	0.948148	0.97037	0.962963	0.896296	0.955556	0.888889	0.955556
train_accuracy	0.940447	1.0	1.0	0.923077	0.975186	0.905707	1.0
precision	0.983871	1.0	1.0	0.873239	1.0	0.851351	1.0
recall	0.910448	0.940299	0.925373	0.925373	0.910448	0.940299	0.910448
f1	0.945736	0.969231	0.96124	0.898551	0.953125	0.893617	0.953125
conf_matrix	[[67, 1], [6, 61]]	[[68, 0], [4, 63]]	[[68, 0], [5, 62]]	[[59, 9], [5, 62]]	[[68, 0], [6, 61]]	[[57, 11], [4, 63]]	[[68, 0], [6, 61]]
auc_roc	(0.9478709394205443,)	(0.9701492537313433,)	(0.9626865671641791,)	(0.8965100965759438,)	(0.9552238805970149,)	(0.8892669007901668,)	(0.9552238805970149,)

As You can see that the SVM is the best with accuracy of 0.99 and yields high recall and precision. And the difference between train_accuracy and accuracy is pretty small which means models aren't overfitting.

Here for Lung Cancer detection, I think even though the difference between false positive and false negative can’t be huge. But in the health related model the precision is more important otherwise we will classify the people who don't have disease as having one.

For the detailed implementation of other models please check the jupyter notebook.

This is the model implementation

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

```
from sklearn.model_selection import RandomizedSearchCV
param_grid={'C':[0.001,0.01,0.1,1,10,100],
'gamma':[0.001,0.01,0.1,1,10,100]}
rcv=RandomizedSearchCV(SVC(),param_grid,cv=5)
rcv.fit(X_train,y_train)
```

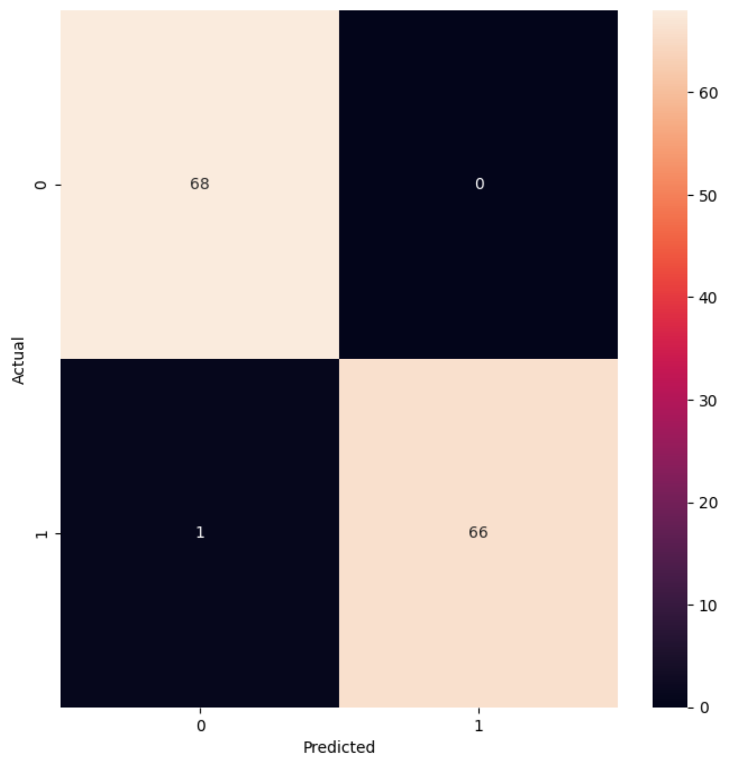
```
predTrain,pred =  getPredPredTrain(rcv)
print(predTrain.shape, pred.shape, y_test.shape,y_train.shape)
metrics = getMetrics(y_test, pred,predTrain,y_train)
raw_data_Modles['SVM'] = metrics
```

Here created the SVM model using RandomizedSearchCV and trained. Let's calculate the important features of this model.

```
from sklearn.inspection import permutation_importance
```

```
result = permutation_importance(rcv, X_train, y_train, n_repeats=10,
random_state=42)
important_features = result.importances_mean
important_features_dict = {}
for i in range(len(important_features)):
  important_features_dict[X_train.columns[i]] = important_features[i]

# important_features_dict
important_features_dict = dict(sorted(important_features_dict.items(),
key=lambda x: x[1]))  # Sorting by values
important_features_dict
```

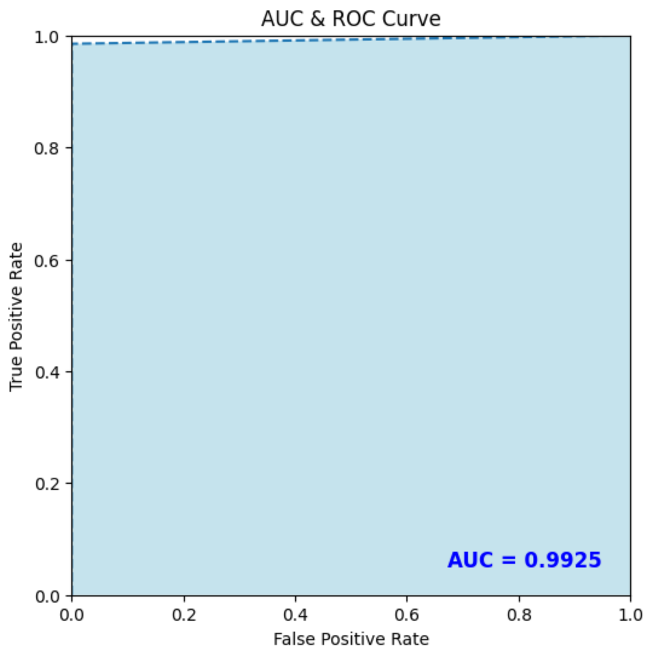
```
{'ALLERGY ': 0.18833746898263026,
 'SWALLOWING DIFFICULTY': 0.1923076923076923,
 'CHRONIC DISEASE': 0.21116625310173695,
 'ALCOHOL CONSUMING': 0.22655086848635236,
 'WHEEZING': 0.2285359801488834,
 'PEER_PRESSURE': 0.2287841191066997,
 'COUGHING': 0.22878411910669977,
 'SMOKING': 0.23548387096774195,
 'GENDER': 0.24119106699751863,
 'ANXIETY': 0.24764267990074443,
 'FATIGUE ': 0.2491315136476427,
 'CHEST PAIN': 0.250620347394541,
 'YELLOW_FINGERS': 0.2550868486352357,
 'SHORTNESS OF BREATH': 0.256575682382134,
 'AGE': 0.34069478908188583}
                       Confusion Matrix of SVM model
```



Only one misclassified data in the prediction of the model.

ROC Curve of the model

```
from sklearn import metrics
auc = metrics.roc_auc_score(y_test, y_pred_svc)

false_positive_rate, true_positive_rate, thresolds =
metrics.roc_curve(y_test, y_pred_svc)

plt.figure(figsize=(6, 6), dpi=100)
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("AUC & ROC Curve")
plt.plot(false_positive_rate, true_positive_rate,'--' )
plt.fill_between(false_positive_rate, true_positive_rate,
facecolor='lightblue', alpha=0.7)
plt.text(0.95, 0.05, 'AUC = %0.4f' % auc, ha='right', fontsize=12,
weight='bold', color='blue')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```



Above the roc curve of the model which has a value of 0.9925 and implementation to evaluate the model.

Discussion & Conclusion

The SVM model result is as follows.

```
'accuracy': 0.9925925925925926,
'train_accuracy': 1.0,
'precision': 1.0,
'recall': 0.9850746268656716,
'f1': 0.9924812030075187,
'conf_matrix': array([[68,  0],
                      [ 1, 66]]),
'auc_roc': (0.9925373134328358,
```

The model achieved best performance among all the trained models and the difference between accuracy and train_accuracy is not high so no overfitting and also precision and recall is also high which is so critical for health related models in order to reduce false positives and false negatives.

Future Improvement

SVM has already achieved best but for the second best model RFC (Random Forest Classifier) we can make the model more improved by turning the hyperparameter and adding more data which faster to compute than SVM on large dataset more robust to imbalance dataset which usually come along with more data.