Ensemble Techniques for Lung Cancer Detection using Machine Learning and PCA

In this research study, we tried to predict Lung Cancer using 4 different algorithm:

SVM (Support Vector Machine) classification Decision tree classification Random forest classification K-Nearest Neighbor classification Predictor variable use in classifying lung cancer:

Age Smokes AreaQ Alkhol

Import Library

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Access data set

In [2]:
```
data = pd.read_csv('lung_cancer_patients_dataset.csv')
print('Dataset :',data.shape)
data.info()
data[0:10]
```

```
Dataset : (59, 7)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Name     59 non-null     object
 1   Surname  59 non-null     object
 2   Age      59 non-null     int64
 3   Smokes   59 non-null     int64
 4   AreaQ    59 non-null     int64
 5   Alkhol   59 non-null     int64
 6   Result   59 non-null     int64
dtypes: int64(5), object(2)
memory usage: 3.4+ KB
```

Out[2]:

	Name	Surname	Age	Smokes	AreaQ	Alkhol	Result
0	John	Wick	35	3	5	4	1
1	John	Constantine	27	20	2	5	1
2	Camela	Anderson	30	0	5	2	0
3	Alex	Telles	28	0	8	1	0
4	Diego	Maradona	68	4	5	6	1
5	Cristiano	Ronaldo	34	0	10	0	0

	Name	Surname	Age	Smokes	AreaQ	Alkhol	Result
6	Mihail	Tal	58	15	10	0	0
7	Kathy	Bates	22	12	5	2	0
8	Nicole	Kidman	45	2	6	0	0
9	Ray	Milland	52	18	4	5	1

In [5]:
```
data.describe()
```

Out[5]:

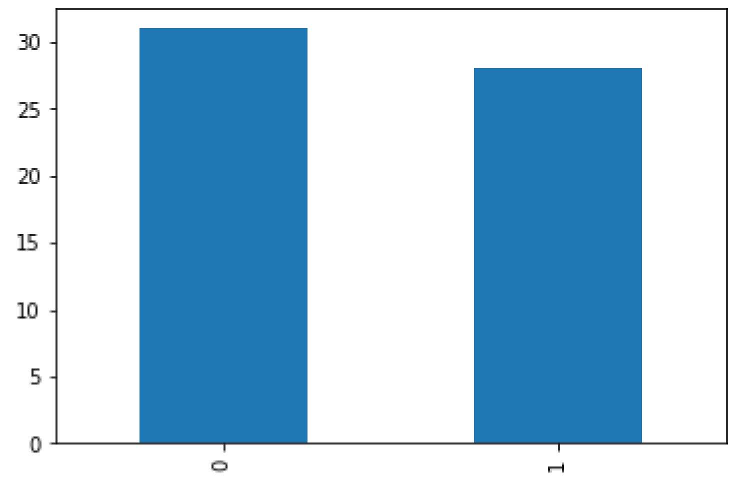
	Age	Smokes	AreaQ	Alkhol	Result
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	42.627119	15.067797	5.203390	3.237288	0.474576
std	16.235230	7.984607	2.461984	2.380517	0.503640
min	18.000000	0.000000	1.000000	0.000000	0.000000
25%	29.000000	10.000000	3.000000	1.000000	0.000000
50%	39.000000	15.000000	5.000000	3.000000	0.000000
75%	55.000000	20.000000	7.500000	5.000000	1.000000
max	77.000000	34.000000	10.000000	8.000000	1.000000

In [6]:
```
data.isna().sum()
```

Out[6]:
```
Name       0
Surname    0
Age        0
Smokes     0
AreaQ      0
Alkhol     0
Result     0
dtype: int64
```

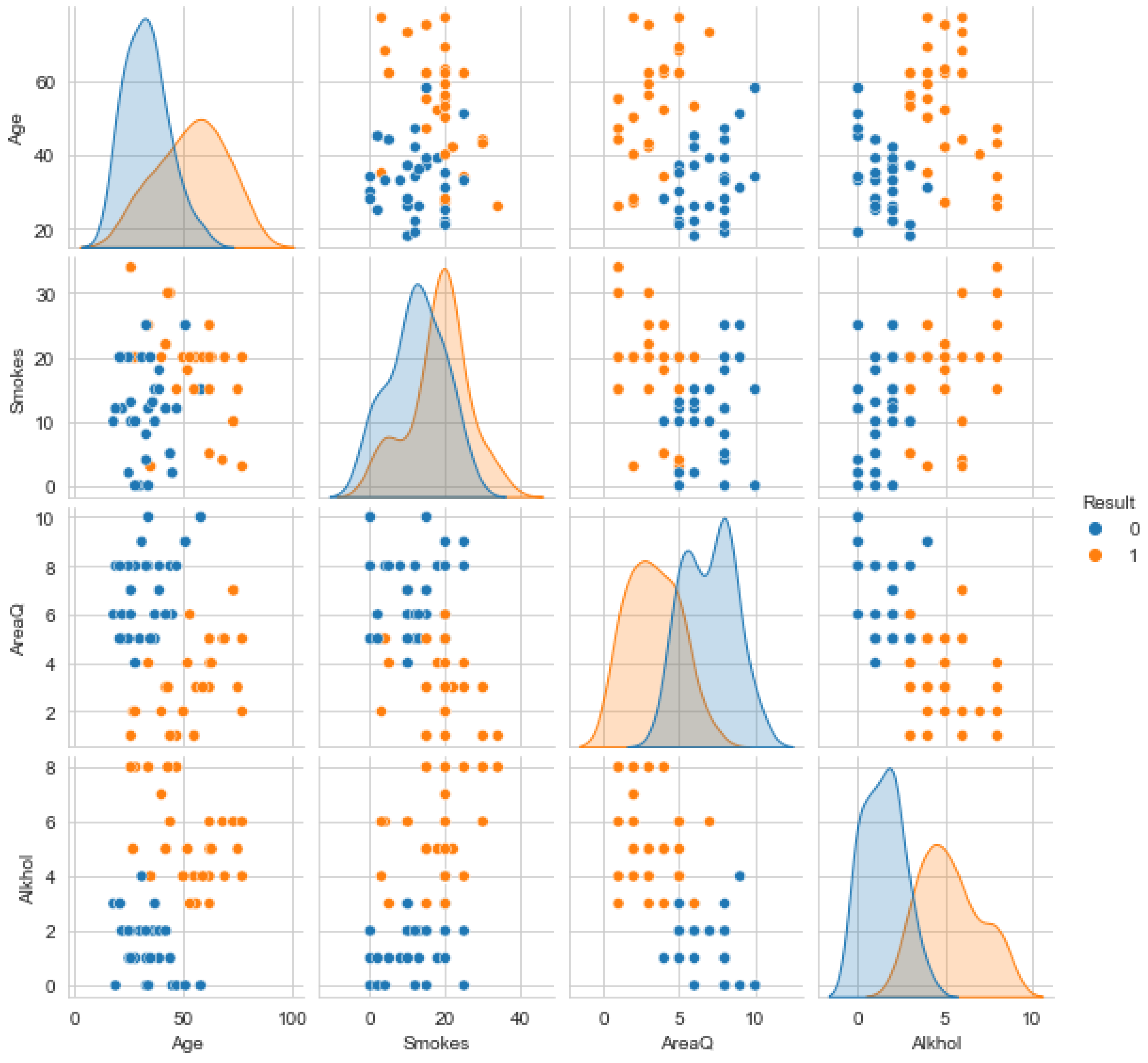
Scaling of data

In [7]:
```
# Distribution of diagnosis
data.Result.value_counts()[0:30].plot(kind='bar')
plt.show()
```



```
In [9]:  sns.set_style("whitegrid")
         sns.pairplot(data,hue="Result",size=2);
         plt.show()
```

```
C:\Users\DR_B_LAL\anaconda3\anaconda2021\lib\site-packages\seaborn\axisgrid.py:1912: Use
rWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



```
data=data.drop(columns=['Name','Surname'])
```

```
In [10]: data.head(5)
```

Out[10]:

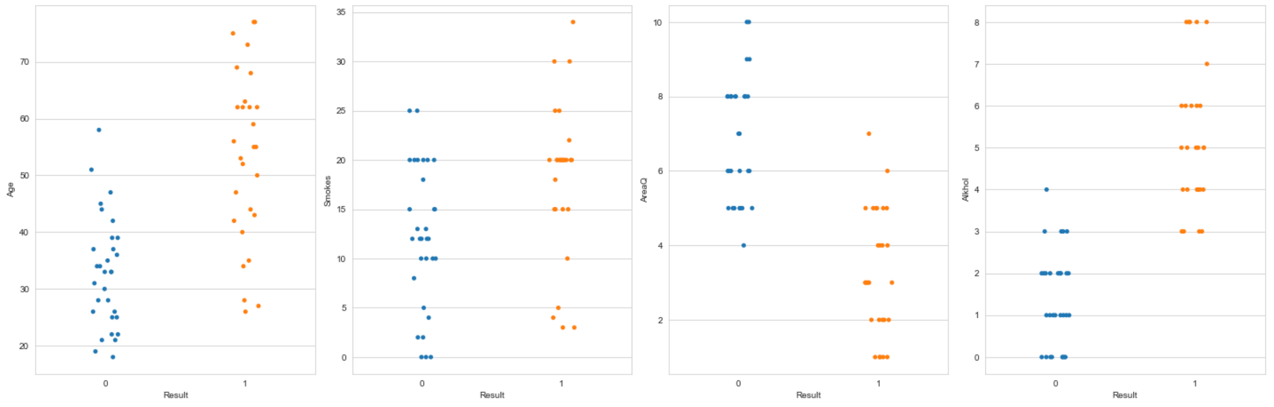
	Age	Smokes	AreaQ	Alkhol	Result
0	35	3	5	4	1
1	27	20	2	5	1
2	30	0	5	2	0
3	28	0	8	1	0
4	68	4	5	6	1

eliminate irrevant data items

```
In [11]: X = data.drop(columns = ['Result'])
         y = data['Result']
```

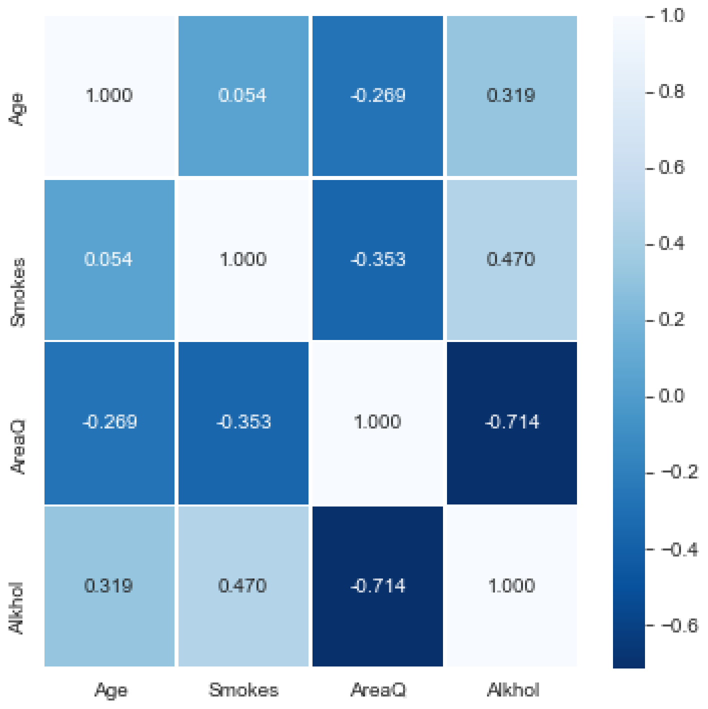
```
In [12]: plt.figure(figsize=(20,25), facecolor='white')
         plotnumber = 1

         for column in X:
             if plotnumber<=16 :
                 ax = plt.subplot(4,4,plotnumber)
                 sns.stripplot(y,X[column])
             plotnumber+=1

         plt.tight_layout()
```

```
C:\Users\DR_B_LAL\anaconda3\anaconda2021\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\DR_B_LAL\anaconda3\anaconda2021\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\DR_B_LAL\anaconda3\anaconda2021\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\DR_B_LAL\anaconda3\anaconda2021\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [13]:
```
corr = X.corr()
f, ax = plt.subplots(figsize = (6,6))
sns.heatmap(corr, annot=True, fmt=".3f", linewidths=0.5,cmap="Blues_r", ax=ax)
```

Out[13]:
```
<AxesSubplot:>
```



	Age	Smokes	AreaQ	Alkhol
Age	1.000	0.054	-0.269	0.319
Smokes	0.054	1.000	-0.353	0.470
AreaQ	-0.269	-0.353	1.000	-0.714
Alkhol	0.319	0.470	-0.714	1.000

scalling of data

split dataset into traning and testing

In [17]:
```
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
x_scaled=scalar.fit_transform(X)
```

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV,train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,roc_curve, roc_auc
```

In [19]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size=0.30,random_state
In [23]: dtc = DecisionTreeClassifier()
         ran = RandomForestClassifier(n_estimators=90)
         knn = KNeighborsClassifier(n_neighbors=41)
         svm = SVC(random_state=6)
```

```
In [24]: models = {"Decision tree" : dtc,
                   "Random forest" : ran,
                   "KNN" : knn,
                   "SVM" : svm}
         scores= { }
```

```
In [25]: for key, value in models.items():
             model = value
             model.fit(x_train, y_train)
             scores[key] = model.score(x_test, y_test)
```

```
In [26]: scores_frame = pd.DataFrame(scores, index=["Accuracy Score"]).T
         scores_frame.sort_values(by=["Accuracy Score"], axis=0 ,ascending=False, inplace=True)
         scores_frame
```

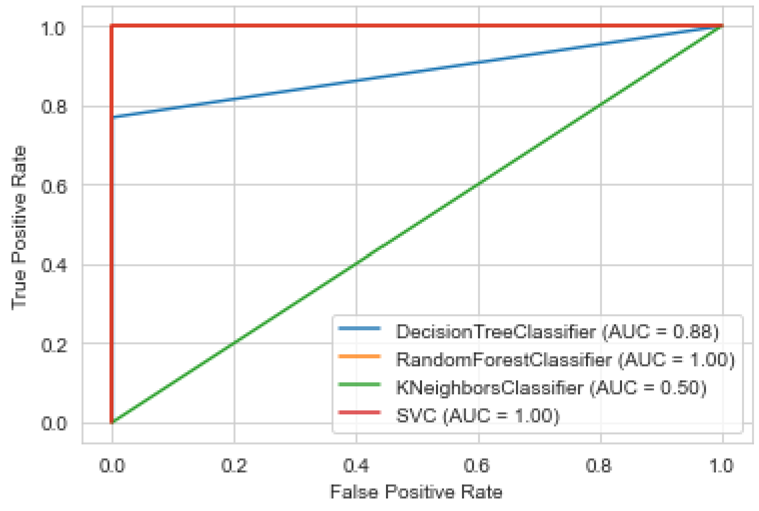
Out[26]:

	Accuracy Score
Random forest	0.944444
SVM	0.944444
Decision tree	0.833333
KNN	0.277778

```
In [27]: from sklearn.metrics import plot_roc_curve
```

```
In [28]: disp = plot_roc_curve(dtc,x_test,y_test)

         plot_roc_curve(ran,x_test,y_test,ax=disp.ax_)

         plot_roc_curve(knn,x_test,y_test,ax=disp.ax_)

         plot_roc_curve(svm,x_test,y_test,ax=disp.ax_)
```

```
Out[28]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1aee10132b0>
```



Let's evaluate with other metrics

```
In [29]: predicted_svc=svm.predict(x_test)
```

```
In [30]: predicted_knn=knn.predict(x_test)
```

Evaluation of SVC

```
In [31]: accuracy=accuracy_score(y_test,predicted_svc)
         print("The accuracy of svc model is : ",accuracy)
```

```
The accuracy of svc model is :  0.9444444444444444
```

```
In [32]: conf_mat = confusion_matrix(y_test,predicted_svc)
         print("The Confusion Matrix for SVC in this dataset is : \n",conf_mat)
```

```
The Confusion Matrix for SVC in this dataset is : 
 [[ 5  0]
 [ 1 12]]
```

```
In [33]: true_positive = conf_mat[0][0]
         false_positive = conf_mat[0][1]
         false_negative = conf_mat[1][0]
         true_negative = conf_mat[1][1]
```

```
In [34]: # Precison
         Precision = true_positive/(true_positive+false_positive)
         print("The precision of this svc model is : ",Precision)
```

```
The precision of this svc model is :  1.0
```

```
In [35]: # Recall
         Recall= true_positive/(true_positive+false_negative)
         print("The Recall score of svc model is : ",Recall)
```

```
The Recall score of svc model is :  0.8333333333333334
```

```
In [36]: F1_Score = 2*(Recall * Precision) / (Recall + Precision)
         print("The F1_Score for this dataset is : ",F1_Score)
```

```
The F1_Score for this dataset is :  0.9090909090909091
```

Evaluation of KNN

```
In [37]: accuracy=accuracy_score(y_test,predicted_knn)
         print("The accuracy of knn model is : ",accuracy)
```

```
The accuracy of knn model is :  0.2777777777777778
```

```
In [38]: conf_mat = confusion_matrix(y_test,predicted_knn)
         print("The Confusion Matrix for KNN in this dataset is : \n",conf_mat)
```

```
The Confusion Matrix for KNN in this dataset is : 
 [[ 5  0]
 [13  0]]
```

```
In [39]: true_positive = conf_mat[0][0]
         false_positive = conf_mat[0][1]
         false_negative = conf_mat[1][0]
         true_negative = conf_mat[1][1]
```

In [40]:
```
# Precison
Precision = true_positive/(true_positive+false_positive)
print("The precision of this knn model is : ",Precision)
```

```
The precision of this knn model is :  1.0
```

In [41]:
```
# Recall
Recall= true_positive/(true_positive+false_negative)
print("The Recall score of knn model is : ",Recall)
```

```
The Recall score of knn model is :  0.2777777777777778
```

In [42]:
```
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
print("The F1_Score for this dataset is : ",F1_Score)
```

```
The F1_Score for this dataset is :  0.4347826086956522
```

Conclusion

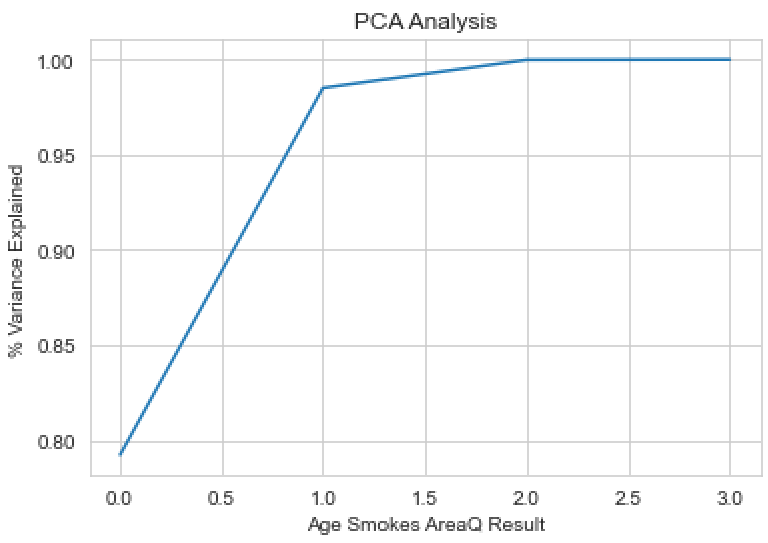
SVC gives a better result than other models,in terms of Accuracy score,Auc score and F1_score Svc gives good result. so we can take svc to predict whether a person has lung cancer or not with good accuracy of 94%.

PCA Analysis

In [44]:
```
Y1 = data['Result']
X1 = data.drop(columns=['Alkhol'])
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel

lsvc = LinearSVC(C=0.06, penalty="l1", dual=False,random_state=10).fit(X1, Y1)
model = SelectFromModel(lsvc, prefit=True)
X_new = model.transform(X1)
cc = list(X1.columns[model.get_support(indices=True)])
print(cc)
print(len(cc))
```

```
['Age', 'Smokes', 'AreaQ']
3
```

In [45]:
```
# Principal component analysis
from sklearn.decomposition import PCA

pca = PCA().fit(X1)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Age Smokes AreaQ Result')
plt.ylabel('% Variance Explained')
plt.title('PCA Analysis')
plt.grid(True)
plt.show()
```



```
In [46]: # Percentage of total variance explained
         variance = pd.Series(list(np.cumsum(pca.explained_variance_ratio_)),
                                          index= list(range(1,5)))
         print(variance[10:90])

         Series([], dtype: float64)

In [ ]:
```