DEEP LEARNING USING TENSORFLOW

ASSIGNMENT – 2

LUNG CANCER DETECTION USING DEEP LEARNING

ABSTRACT:

Lung cancers have been identified as one of the world's most serious causes of death. It is among the most malignant tumors that can affect human wellbeing. Its death rate scores among all tumor deaths, and is also the top killer towards male and female cancer death. There have been nearly 1.8 million fresh cases of lung cancer annually (13 percent among all cancers), 1.6 million deaths worldwide (19.4 percent among all cancers). Lung cancer is a proliferation of expanding and developing irregular cells into a tumour. Of the other forms of cancer, the death rate of lung cancer is the greatest. Cigarette smoke induces an approximate 85 percent of cases of lung cancer in males and 75 percent in females. Lung cancer is amongst the most terrible illnesses in the developing countries, with a death rate of 19.4 percent. Lung cancer is among the most dangerous cancer worldwide, with lowest success rate following diagnosis, with a steady rise in casualty count per year. Here we proposed a deep learning model using Logistic Regression, Naïve Bayes, SVM, KNN to detect lung cancer using supervised learning.

LOGISTIC REGRESSION:

Logistic regression is a process of modelling the probability of a discrete outcome given an input variable. It is used in statistical software to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation. This type of analysis can help you predict the likelihood of an event happening or a choice being made.

SVM:

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are: Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples.

NAIVE BAYES:

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

KNN:

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

SUPERVISED LEARNING:

Supervised learning is an approach to creating artificial intelligence (AI), where a computer algorithm is trained on input data that has been labelled for a particular output.

PROGRAM CODE: (PYTHON)

```
# -*- coding: utf-8 -*-

"""Hons_Assignment.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/17QLh2Zz0rszWScS1PiHQ7hHGnd5LlZ_Q#
Commented out IPython magic to ensure Python compatibility.

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# %matplotlib inline

df=pd.read_csv('lung_cancer_examples.csv')

df.head()


df.shape


df.describe()


df.info


df["Smokes"].describe()


df["Smokes"].unique()
```

```
array([ 3, 20,  0,  4, 15, 12,  2, 18, 10, 25, 22,  5,  8, 13, 30, 34])
```

```
print(df.corr()["Smokes"].abs().sort_values(ascending=False))
```

```
Smokes    1.000000
```

```
Alkhol    0.469915
Result    0.373444
AreaQ     0.353295
Age       0.053665
Name: Smokes, dtype: float64
```

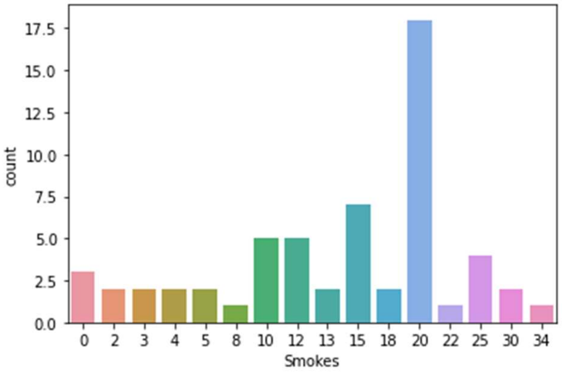
y = df["Smokes"]

sns.countplot(y)

target_temp = df.Smokes.value_counts()

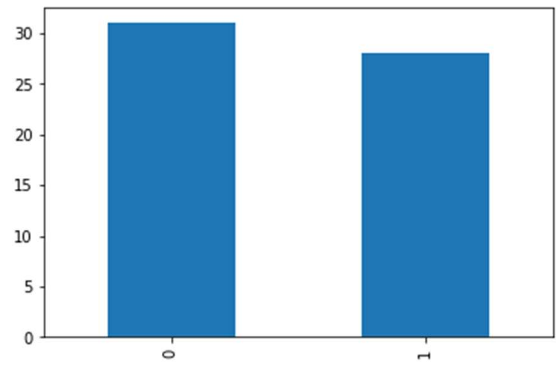
print(target_temp)

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error
or misinterpretation.
  FutureWarning
20    18
15     7
12     5
10     5
25     4
0      3
3      2
4      2
2      2
18     2
5      2
13     2
30     2
22     1
8      1
34     1
Name: Smokes, dtype: int64
```

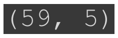


df.Result.value_counts()[0:30].plot(kind='bar')

plt.show()

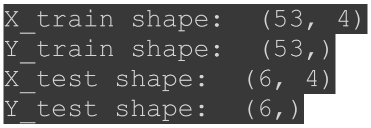


```
df1 = df.drop(columns=['Name','Surname'],axis=1)

df1 = df1.dropna(how='any')

print(df1.shape)
```



```
(59, 5)
```

```
print(df1.shape)

df1.head()


from sklearn.model_selection import train_test_split

Y = df1['Result']

X = df1.drop(columns=['Result'])

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, random_state=9)


print('X_train shape: ', X_train.shape)

print('Y_train shape: ', Y_train.shape)

print('X_test shape: ', X_test.shape)

print('Y_test shape: ', Y_test.shape)
```

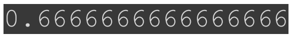


```
X_train shape:  (53, 4)
Y_train shape:  (53,)
X_test shape:  (6, 4)
Y_test shape:  (6,)
```

```
"""Using Logistic Regression,"""


from sklearn.linear_model import LogisticRegression
```

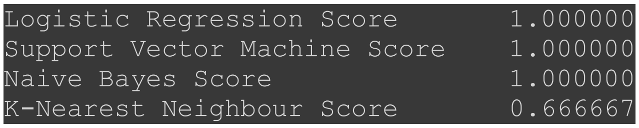
```
logreg = LogisticRegression(C=10)
logreg.fit(X_train, Y_train)
Y_predict1 = logreg.predict(X_test)

Accuracy_1 = logreg.score(X_test, Y_test)
print(Accuracy_1)
1.0

"""Using Support Vector Machines(SVM),"""

from sklearn.ensemble import BaggingClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
svm = OneVsRestClassifier(BaggingClassifier(SVC(C=10,kernel='rbf',random_state=9,
probability=True),n_jobs=-1))
svm.fit(X_train, Y_train)
Y_predict2 = svm.predict(X_test)

Accuracy_2 = svm.score(X_test, Y_test)
print(Accuracy_2)
1.0

"""Using Naive Bayes Classification,"""

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, Y_train)
pred = nb.predict(X_test)

Accuracy_3= nb.score(X_test, Y_test)
print(Accuracy_3)
1.0
```

```
"""Using K Nearest Neighbor (KNN),"""

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5,n_jobs=-1)
knn.fit(X_train, Y_train)
pred1 = knn.predict(X_test)

Accuracy_4= knn.score(X_test, Y_test)
print(Accuracy_4)
```



```
0.6666666666666666
```

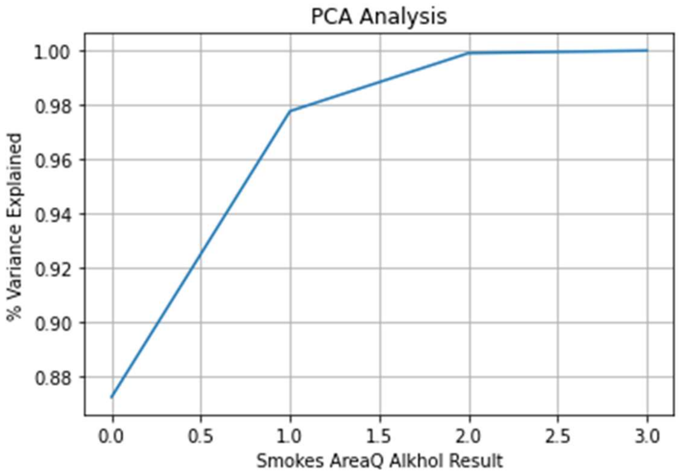
```
Test_Acc = pd.Series([Accuracy_1, Accuracy_2, Accuracy_3,Accuracy_4], index=['Logistic Regression Score', 'Support Vector Machine Score', 'Naive Bayes Score', 'K-Nearest Neighbour Score'])
print(Test_Acc)
```



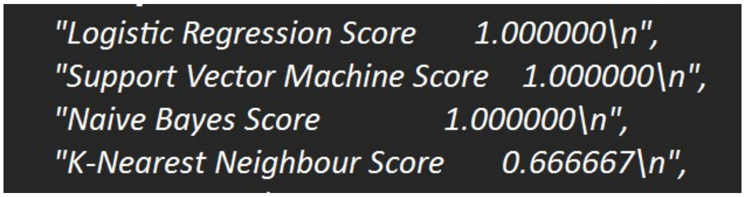
```
Logistic Regression Score       1.000000
Support Vector Machine Score    1.000000
Naive Bayes Score               1.000000
K-Nearest Neighbour Score       0.666667
```

```
dtype: float64
```

```
from sklearn.decomposition import PCA
import numpy as np

X1 = df1.drop(columns=['Age'])
pca = PCA().fit(X1)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Smokes AreaQ Alkhol Result')
plt.ylabel('% Variance Explained')
plt.title('PCA Analysis')
plt.grid(True)
plt.show()
```



OUTPUT:



```
"Logistic Regression Score      1.000000\n",
"Support Vector Machine Score   1.000000\n",
"Naive Bayes Score              1.000000\n",
"K-Nearest Neighbour Score      0.666667\n",
```

CONCLUSION:

Except KNN, remaining Logistic Regression, SVM, Naive Bayes performed well.