Title of the Project: Emotion Recognition Using Speech

Introduction and Objectives of the Project: Emotion recognition using speech is a field of study that deals with the detection and analysis of human emotions from speech signals. The goal of this project is to develop a system that can automatically identify emotional states from speech using machine learning techniques. The system will be able to recognize a range of emotions, including happiness, sadness, anger, fear, and disgust.

Project Category: Artificial Intelligence and data analysis

Tools/Platform, Hardware and Software Requirements: The project will be implemented using Python programming language and several libraries such as Scikit-learn, Tensorflow, and Keras for machine learning. The hardware requirements include a computer with a processor of at least 2 GHz, 8 GB RAM, and a minimum of 250 GB hard disk space.

Problem Definition, Requirement Specifications, Project Planning and Scheduling: The problem is to develop a machine learning model that can accurately recognize emotions from speech signals. The requirements include collecting a large and diverse dataset of annotated speech signals, preprocessing the data, extracting features, and training a model using supervised learning techniques. The project will be planned and scheduled using a Gantt chart and PERT graph to ensure timely completion.

Scope of the arrangement: The scope of the project includes collecting a dataset of speech signals, preprocessing the data, feature extraction, model development, and evaluation. The project will focus on recognizing five basic emotions: happiness, sadness, anger, fear, and disgust.

Analysis: The data models for the project include 0, 1, and 2 level DFDs, complete ER diagrams with cardinality, and class diagrams. The data models will be used to analyze the data and identify the relationships between the different entities in the system.

Database and Tables Detail: The database for the project will include tables for storing the speech signals, feature vectors, and the emotional labels. The primary keys for the tables will be the unique identifiers for each speech signal, and the foreign keys will be used to establish relationships between the tables. The tables will also have appropriate constraints on the fields, such as data type, length, and nullability, according to the project requirements.

This code is for performing various machine learning classification algorithms on a dataset containing emotional speech recordings. The code imports various libraries such as numpy, pandas, tensorflow, seaborn, and matplotlib.pyplot. It also sets a warning filter to ignore future warnings.

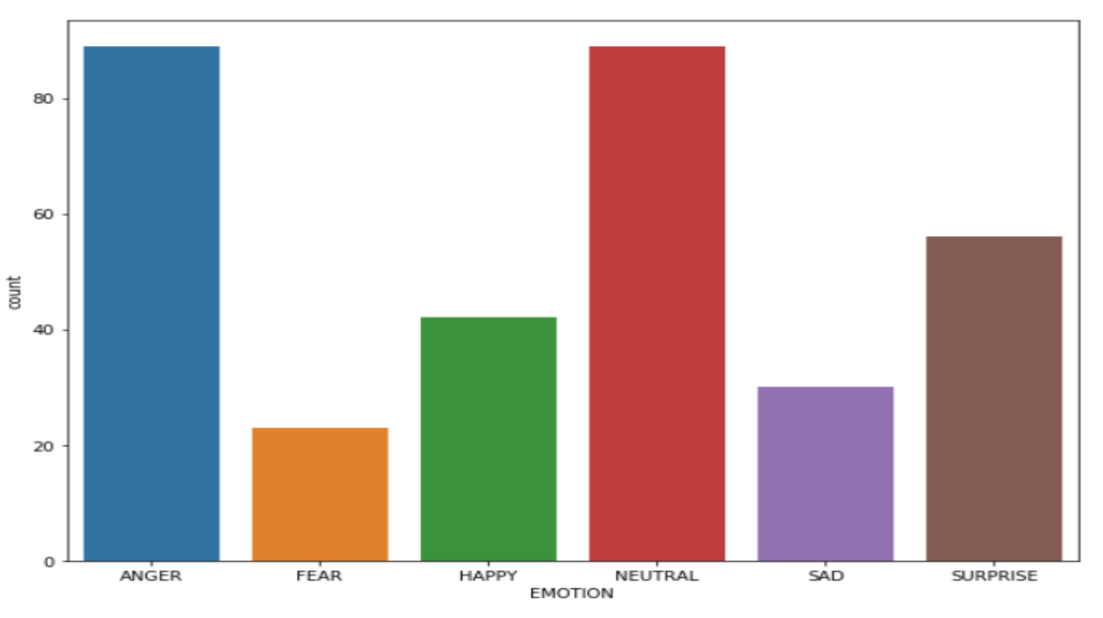
The code then reads in a .csv file and saves it to a Pandas dataframe variable df. The dataframe is explored through various methods like head(), info(), corr(), unique(), and isnull().sum().sum(). The EMOTION column of the dataframe is the target variable and the other columns are the input features.

The code then performs various classification algorithms like logistic regression, random forest, gradient boosting, and support vector machines (SVM). It also performs some preprocessing steps like one-hot encoding and PCA (Principal Component Analysis). For each algorithm, it first performs a grid search to find the best hyperparameters, fits the model on the training data, predicts on the test data, and evaluates the performance using classification_report and confusion_matrix functions.

Finally, the code visualizes the confusion matrices using sns.heatmap() function for each algorithm's predictions.

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

import os
print(os.listdir("D:\DNO\major"))
for df in ("D:\DNO\major"):
    df=pd.read_csv("D:\DNO\major/preprocessing.csv").fillna(0)

# Any results you write to the current directory are saved as output.
df.head()
df.info()
df.corr()
df['EMOTION'].unique()
plt.figure(figsize = (10, 8))
sns.countplot(df['EMOTION'])
```

```
plt.show()
```

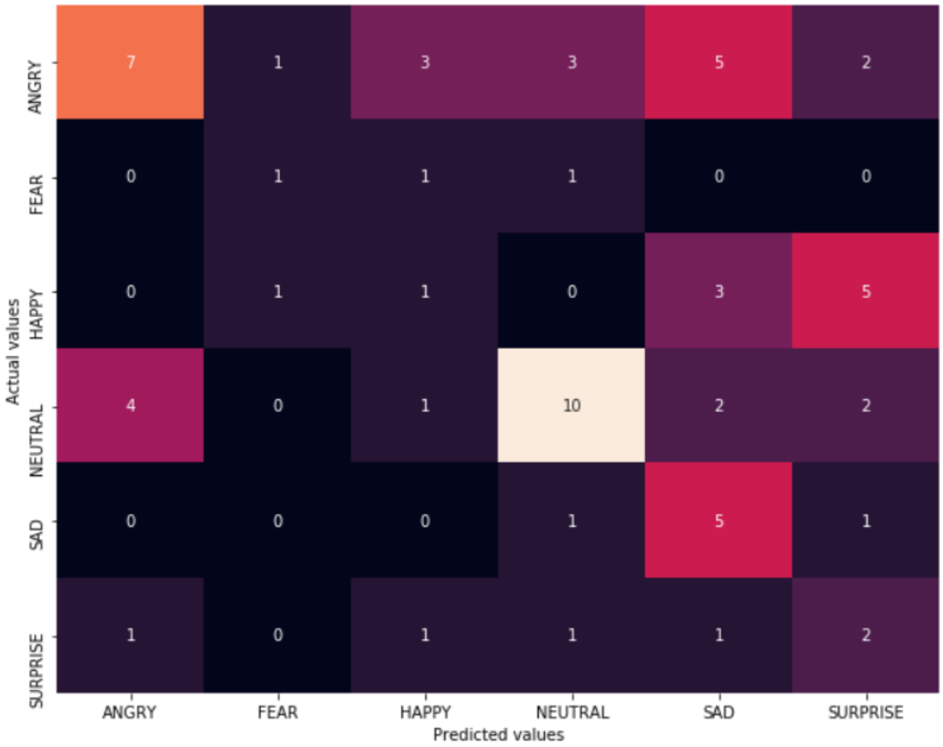


```
df['EMOTION'].value_counts()
df.isnull().sum().sum() #no missing values
#split into features and labels sets
X = df.drop(['EMOTION','ID'], axis = 1) #features
y = df['EMOTION'] #labels

X.head()

X.info()

print("Total number of labels: {}".format(df.shape[0]))

target = df.ID

X.dtypes.sample(104)

one_hot_encoded_training_predictors = pd.get_dummies(X)
one_hot_encoded_test_predictors = pd.get_dummies(y)
final_train, final_test = one_hot_encoded_training_predictors.align(one_hot_encoded_test_predictors,join='left', axis=1)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 0)

from sklearn.linear_model import LogisticRegression
m1 = LogisticRegression()
m1.fit(X_train, y_train)
pred1 = m1.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
```

```
print(classification_report(y_test, pred1))
```

```
labels = ['ANGRY','FEAR','HAPPY','NEUTRAL','SAD','SURPRISE']
cm1 = pd.DataFrame(confusion_matrix(y_test, pred1), index = labels, columns = labels)
```

```
plt.figure(figsize = (10, 8))
sns.heatmap(cm1, annot = True, cbar = False, fmt = 'g')
plt.ylabel('Actual values')
plt.xlabel('Predicted values')
plt.show()
```



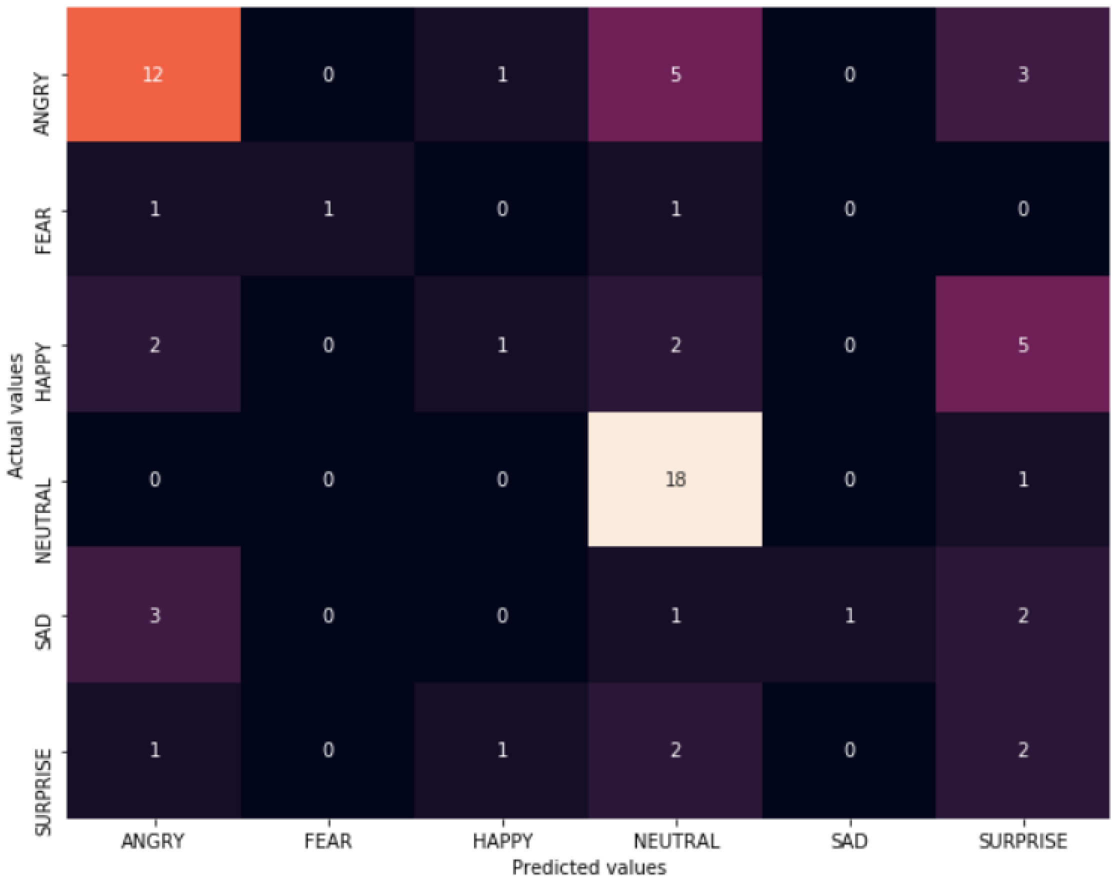
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

```
grid = {'n_estimators': [10, 50, 100, 300]}

m2 = GridSearchCV(RandomForestClassifier(), grid)
m2.fit(X_train, y_train)
```

```
m2.best_params_  #I got n_estimators = 300
```

```
             precision    recall  f1-score   support

      ANGER       0.63      0.57      0.60        21
       FEAR       1.00      0.33      0.50         3
      HAPPY       0.33      0.10      0.15        10
    NEUTRAL       0.62      0.95      0.75        19
        SAD       1.00      0.14      0.25         7
   SURPRISE       0.15      0.33      0.21         6

avg / total       0.60      0.53      0.50        66
```

```
cm2 = pd.DataFrame(confusion_matrix(y_test, pred2), index = labels, columns = labels)

plt.figure(figsize = (10, 8))
sns.heatmap(cm2, annot = True, cbar = False, fmt = 'g')
plt.ylabel('Actual values')
plt.xlabel('Predicted values')
plt.show()
```



```
from sklearn.ensemble import GradientBoostingClassifier
```

```
grid = {
    'learning_rate': [0.03, 0.1, 0.5],
    'n_estimators': [100, 300],
    'max_depth': [1, 3, 9]
}

m3 = GridSearchCV(GradientBoostingClassifier(), grid, verbose = 2)
m3.fit(X_train, y_train)
```

```
m3.best_params_
```

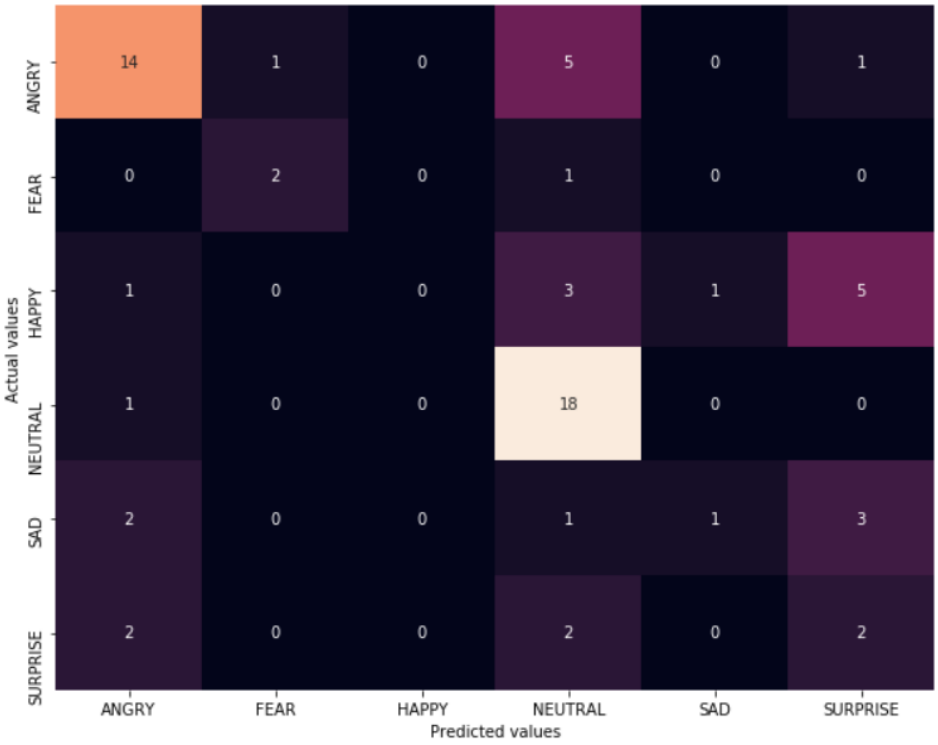
```
{'learning_rate': 0.03, 'max_depth': 1, 'n_estimators': 100}
```

```
pred3 = m3.predict(X_test)
```

```
print(classification_report(y_test, pred3))
```

```
             precision    recall  f1-score   support

      ANGER       0.70      0.67      0.68        21
       FEAR       0.67      0.67      0.67         3
      HAPPY       0.00      0.00      0.00        10
    NEUTRAL       0.60      0.95      0.73        19
        SAD       0.50      0.14      0.22         7
   SURPRISE       0.18      0.33      0.24         6

avg / total       0.50      0.56      0.50        66
```

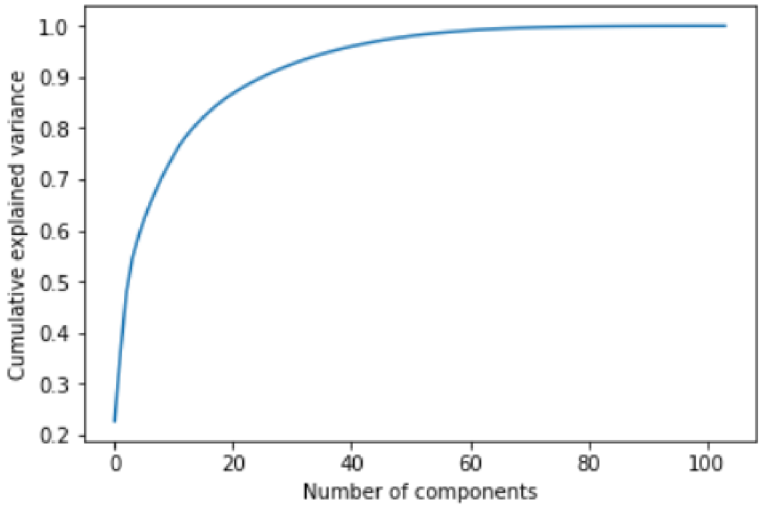
```
cm3 = pd.DataFrame(confusion_matrix(y_test, pred3), index = labels, columns = labels)

plt.figure(figsize = (10, 8))
sns.heatmap(cm3, annot = True, cbar = False, fmt = 'g')
plt.ylabel('Actual values')
plt.xlabel('Predicted values')
plt.show()
```



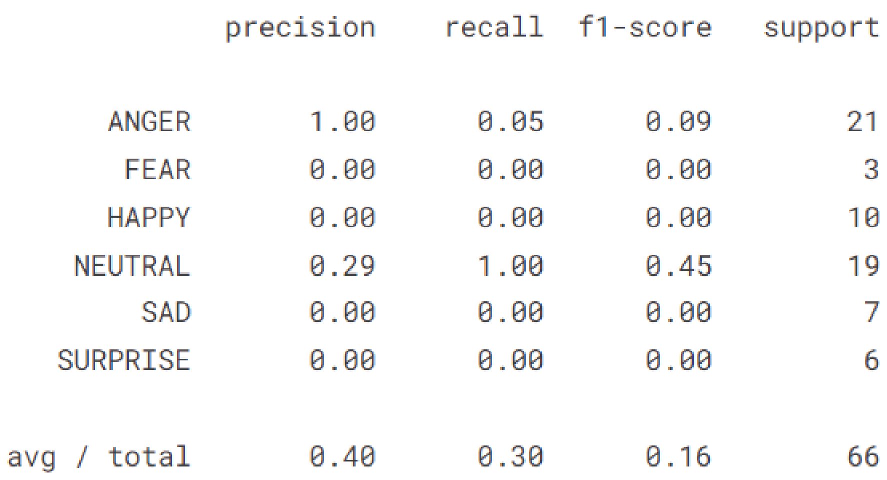
	ANGRY	FEAR	HAPPY	NEUTRAL	SAD	SURPRISE
ANGRY	14	1	0	5	0	1
FEAR	0	2	0	1	0	0
HAPPY	1	0	0	3	1	5
NEUTRAL	1	0	0	18	0	0
SAD	2	0	0	1	1	3
SURPRISE	2	0	0	2	0	2

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
scaler = StandardScaler()
scaler.fit(X_train)
```

```
X_sc_train = scaler.transform(X_train)
X_sc_test = scaler.transform(X_test)

pca = PCA(n_components=104)
pca.fit(X_train)

plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
```

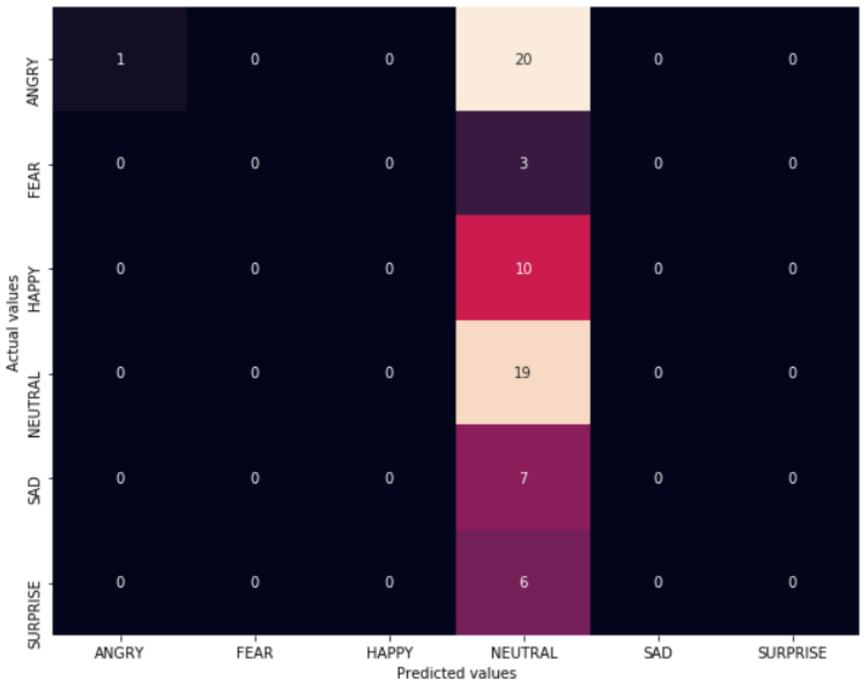


```
NCOMPONENTS = 104

pca = PCA(n_components=NCOMPONENTS)
X_pca_train = pca.fit_transform(X_sc_train)
X_pca_test = pca.transform(X_sc_test)
pca_std = np.std(X_pca_train)

print(X_sc_train.shape)
print(X_pca_test.shape)


inv_pca = pca.inverse_transform(X_pca_train)
inv_sc = scaler.inverse_transform(inv_pca)

grid = {
    'C': [1,5,50],
    'gamma': [0.05,0.1,0.5,1,5]
}

m5 = GridSearchCV(SVC(), grid)
m5.fit(X_train, y_train)


m5.best_params_ #I got C = 1, gamma = 0.05

pred5 = m5.predict(X_test)

print(classification_report(y_test, pred5))
```



```
             precision    recall  f1-score   support

      ANGER       1.00      0.05      0.09        21
       FEAR       0.00      0.00      0.00         3
      HAPPY       0.00      0.00      0.00        10
    NEUTRAL       0.29      1.00      0.45        19
        SAD       0.00      0.00      0.00         7
   SURPRISE       0.00      0.00      0.00         6

avg / total       0.40      0.30      0.16        66
```

```
cm5 = pd.DataFrame(confusion_matrix(y_test, pred5), index = labels, columns = labels)

plt.figure(figsize = (10, 8))
sns.heatmap(cm5, annot = True, cbar = False, fmt = 'g')
plt.ylabel('Actual values')
plt.xlabel('Predicted values')
plt.show()
```


Actual values
ANGRY
FEAR
HAPPY
NEUTRAL
SAD
SURPRISE
ANGRY
FEAR
HAPPY
NEUTRAL
SAD
SURPRISE
Predicted values