CENTRAL UNIVERSITY OF KARNATAKA

Electronics and Communication Dept

Machine Learning & Expert systems

PROJECT REPORT

(Gender Voice Classifier)

GUIDED BY:-

Dr. Nagaraj.Y

Submitted by:

Satyam Kumar

Reg.No :2021UGECE28

Shaik Mahammad Tarik

Reg.No :2021UGECE30

INDEX

ABSTRACT

- Voice-based gender classification is a fundamental task with significant implications for numerous applications, including speech recognition, virtual assistants, and human-computer interaction systems. This research endeavors to explore and compare the efficacy of various machine learning algorithms in accurately identifying the gender of speakers based on voice characteristics. The study employs a diverse dataset encompassing voice recordings from both male and female speakers, sourced from a wide range of contexts.

- To commence the analysis, the dataset undergoes meticulous preprocessing to extract pertinent features, which serve as input for the machine learning classifiers. These classifiers include popular algorithms such as K-Nearest Neighbors, Decision Trees, Support Vector Machines, and Neural Networks. Each algorithm is trained on the dataset, and its performance is rigorously evaluated using standard metrics such as accuracy, precision, recall, and F1-score.

- Furthermore, the research delves into feature importance analysis to discern the most influential attributes contributing to gender classification accuracy. This analysis sheds light on the underlying patterns and characteristics crucial for distinguishing between male and female voices.

- The findings of this study provide valuable insights into the comparative effectiveness of different machine learning approaches for voice-based gender classification. By elucidating the strengths and weaknesses of each algorithm, the research aids in the development of more robust and efficient gender identification systems. These systems have the potential to enhance the accuracy and reliability of speech recognition technologies, empower virtual assistants to better understand user preferences, and facilitate seamless human-computer interaction experiences.

PROPOSED METHODOLOGY

Data Collection and Preprocessing:

Acquire a diverse dataset comprising voice recordings from both male and female speakers.

Ensure the dataset covers a wide range of ages, accents, languages, and recording conditions to capture variations in voice characteristics.

Preprocess the dataset by removing noise, normalizing audio levels, and extracting relevant features such as pitch, formants, intensity, and duration using signal processing techniques.

Feature Selection and Extraction:

Conduct feature selection to identify the most discriminative attributes for gender classification.

Utilize techniques such as principal component analysis (PCA) or feature importance analysis to prioritize informative features.

Extract the selected features from the preprocessed audio data, creating a feature matrix where each row corresponds to a voice sample and each column represents a feature.

Model Selection and Training:

Choose a variety of machine learning algorithms suitable for classification tasks, including but not limited to:

K-Nearest Neighbors (KNN)

Decision Trees (DT)

Support Vector Machines (SVM)

Random Forests (RF)

Gradient Boosting Machines (GBM)

Convolutional Neural Networks (CNN)

Split the dataset into training and testing sets using techniques like k-fold cross-validation to ensure robust model evaluation.

Train each selected algorithm on the training set using the extracted features and corresponding gender labels.

Optimize hyperparameters using techniques such as grid search or random search to improve model performance.

Model Evaluation and Comparison:

Evaluate the trained models on the testing set using standard performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Compare the performance of different algorithms to identify the most effective ones for voice-based gender classification.

Conduct statistical tests (e.g., paired t-tests) to assess the significance of performance differences between algorithms.

Analysis of Results and Interpretation:

Analyze the results to gain insights into the strengths and weaknesses of each algorithm.

Investigate the impact of feature selection and extraction techniques on classification performance.

Interpret the findings to understand the underlying factors contributing to successful gender classification.

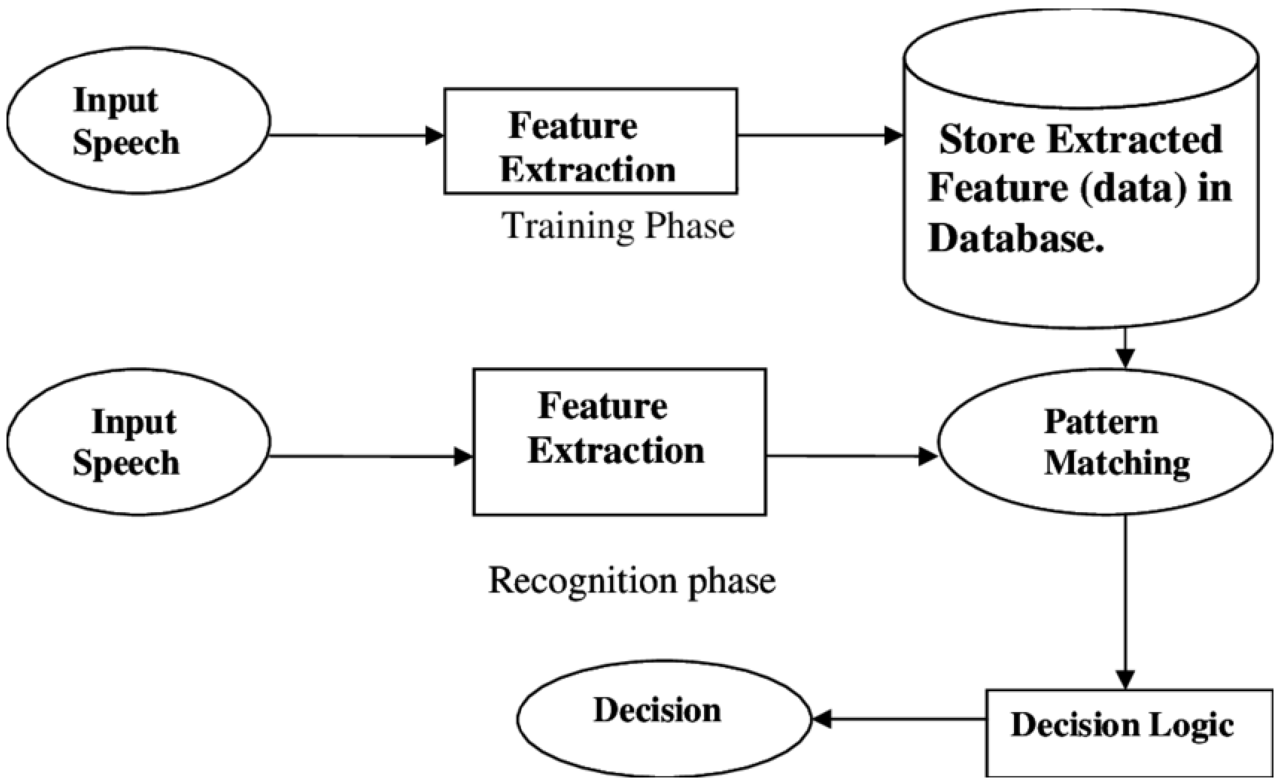
Visualize the results using plots and charts to facilitate clear communication of key findings.

Discussion and Conclusion:

Discuss the implications of the findings for real-world applications such as speech recognition systems, virtual assistants, and human-computer interaction interfaces.

Conclude by summarizing the key contributions of the study

BLOCK DIAGRAM



CODE

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installe
d
# It is defined by the kaggle/python docker image: https://github.com/kaggle/doc
ker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# 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("../input"))

# Any results you write to the current directory are saved as output.
['voice.csv']
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
# read file
voice=pd.read_csv('../input/voice.csv')
voice.head()
```

Out[2]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568

In [3]:

```
voice.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
meanfreq    3168 non-null float64
sd          3168 non-null float64
median      3168 non-null float64
Q25         3168 non-null float64
Q75         3168 non-null float64
IQR         3168 non-null float64
skew        3168 non-null float64
kurt        3168 non-null float64
sp.ent      3168 non-null float64
sfm         3168 non-null float64
mode        3168 non-null float64
centroid    3168 non-null float64
meanfun     3168 non-null float64
minfun      3168 non-null float64
maxfun      3168 non-null float64
meandom     3168 non-null float64
mindom      3168 non-null float64
maxdom      3168 non-null float64
dfrange     3168 non-null float64
modindx     3168 non-null float64
label       3168 non-null object
dtypes: float64(20), object(1)
memory usage: 519.8+ KB
```

In [4]:

```
voice.describe()
```

Out[4]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt
count	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.00
mean	0.180907	0.057126	0.185621	0.140456	0.224765	0.084309	3.140168	36.5684
std	0.029918	0.016652	0.036360	0.048680	0.023639	0.042783	4.240529	134.928
min	0.039363	0.018363	0.010975	0.000229	0.042946	0.014558	0.141735	2.06845
25%	0.163662	0.041954	0.169593	0.111087	0.208747	0.042560	1.649569	5.66954
50%	0.184838	0.059155	0.190032	0.140286	0.225684	0.094280	2.197101	8.31846
75%	0.199146	0.067020	0.210618	0.175939	0.243660	0.114175	2.931694	13.6489
max	0.251124	0.115273	0.261224	0.247347	0.273469	0.252225	34.725453	1309.61

Preprocessing: label encoder and normalization

In [5]:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
voice["label"] = le.fit_transform(voice["label"])
le.classes_
```

Out[5]:

```
array(['female', 'male'], dtype=object)
```

In [6]:

```
voice[:]=preprocessing.MinMaxScaler().fit_transform(voice)
voice.head()
```

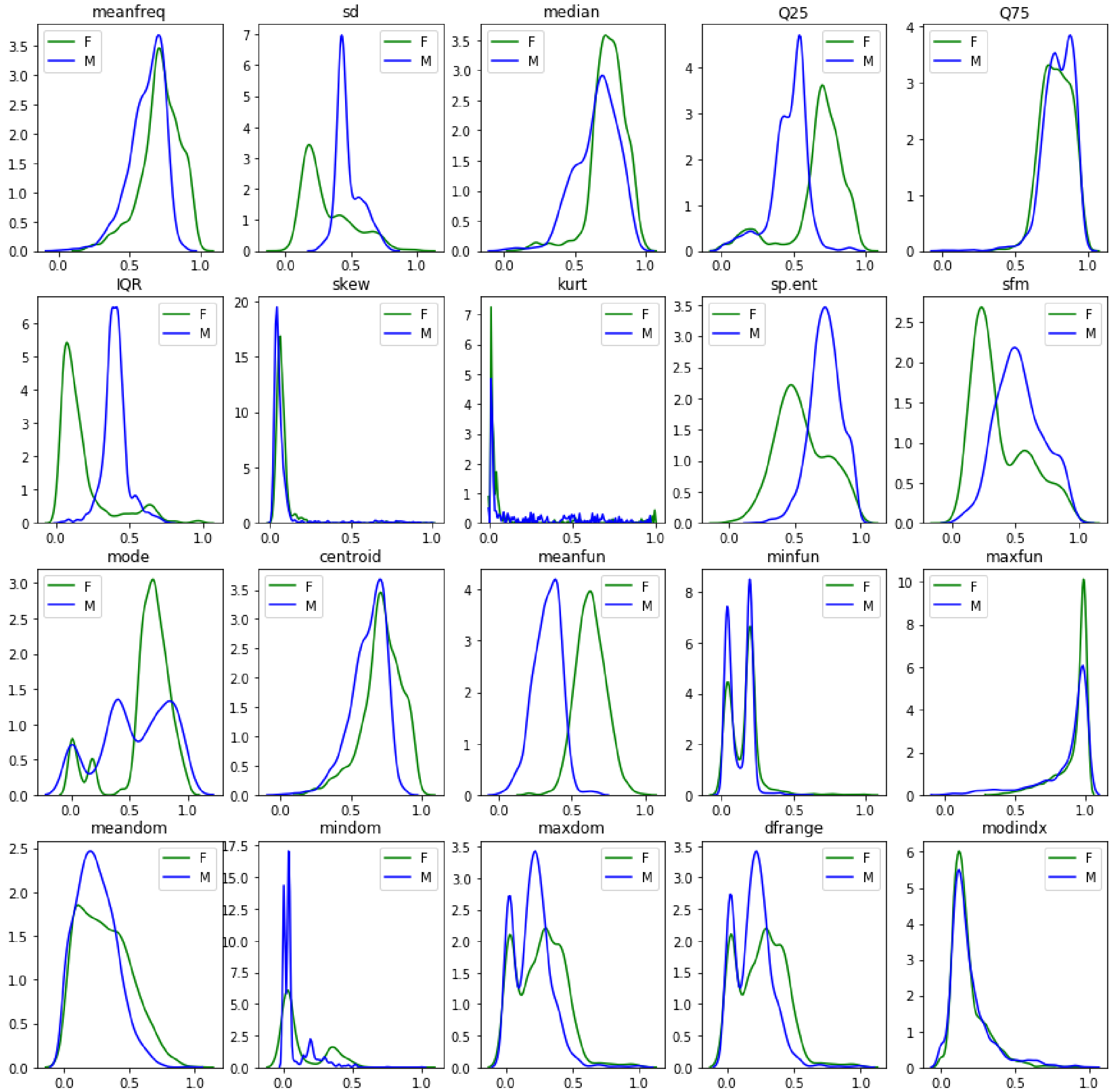
Out[6]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mo
0	0.096419	0.473409	0.084125	0.060063	0.204956	0.254828	0.367853	0.208279	0.635798	0.564526	0.0
1	0.125828	0.505075	0.116900	0.077635	0.215683	0.246961	0.644279	0.483766	0.630964	0.591578	0.0
2	0.179222	0.675536	0.102873	0.034284	0.385912	0.457148	0.885255	0.782275	0.442738	0.548382	0.0
3	0.528261	0.554611	0.587559	0.389906	0.715802	0.407358	0.031549	0.001613	0.923261	0.856457	0.2
4	0.452195	0.627209	0.454272	0.317627	0.707515	0.474474	0.027742	0.001732	0.958736	0.926348	0.3

Visualization

In [7]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.subplots(4,5,figsize=(15,15))
for i in range(1,21):
    plt.subplot(4,5,i)
    plt.title(voice.columns[i-1])
    sns.kdeplot(voice.loc[voice['label'] == 0, voice.columns[i-1]], color= 'gre
en', label='F')
    sns.kdeplot(voice.loc[voice['label'] == 1, voice.columns[i-1]], color= 'blu
e', label='M')
```



At first glance, most significant features are Q25, IQR and meanfun. We will build models by using the 20 features and the 3 distinct features.

Using K-Nearest Neighbors, Naive Bayes, Decision Tree, Random Forest, XgBoost, Support Vector Machine, Neural Network to build models

In [8]:

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn import neighbors
from sklearn import naive_bayes
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
from sklearn import neural_network
import xgboost
```

In [9]:
```
# Split the data
train, test = train_test_split(voice, test_size=0.3)
```

In [10]:
```
train.head()
```

Out[10]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm
454	0.000000	0.434260	0.010441	0.021657	0.000000	0.095966	0.128711	0.019782	0.323817	0.304450
2962	0.610963	0.387141	0.640343	0.588411	0.672090	0.158562	0.058454	0.004755	0.797302	0.652062
1876	0.719393	0.274532	0.717657	0.651222	0.756676	0.175297	0.051910	0.004514	0.707063	0.301095
2593	0.904495	0.153585	0.897756	0.864024	0.910865	0.103587	0.056739	0.004123	0.330340	0.134420
317	0.555580	0.445700	0.575999	0.417268	0.709400	0.372699	0.072282	0.008316	0.765753	0.599947

In [11]:
```
x_train = train.iloc[:, :-1]
y_train = train["label"]
x_test = test.iloc[:, :-1]
y_test = test["label"]
```

In [12]:
```
x_train3 = train[["meanfun","IQR","Q25"]]
y_train3 = train["label"]
x_test3 = test[["meanfun","IQR","Q25"]]
y_test3 = test["label"]
```

In [13]:
```
def classify(model,x_train,y_train,x_test,y_test):
    from sklearn.metrics import classification_report
    target_names = ['female', 'male']
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    print(classification_report(y_test, y_pred, target_names=target_names, digi
ts=4))
```

K-Nearest Neighbors

Using neighbors.KNeighborsClassifier() to build the model.

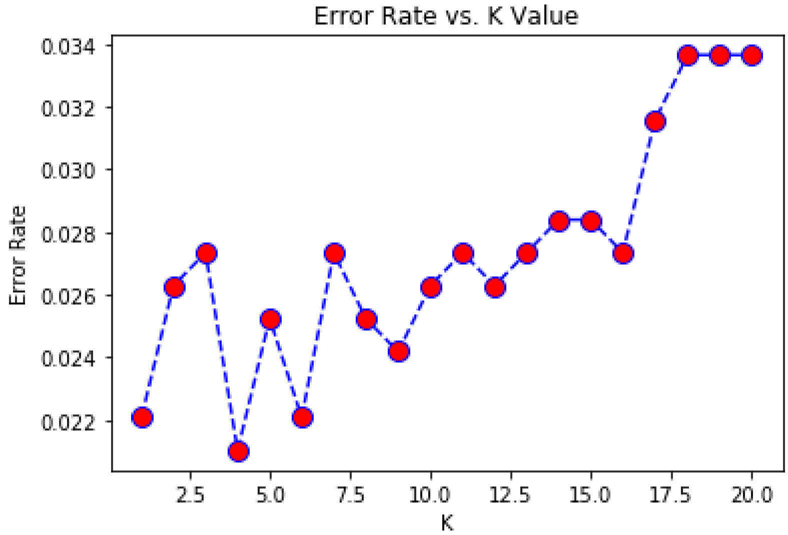
In [14]:
```
def knn_error(k,x_train,y_train,x_test,y_test):
    error_rate = []
    K=range(1,k)
    for i in K:
        knn = neighbors.KNeighborsClassifier(n_neighbors = i)
        knn.fit(x_train, y_train)
        y_pred = knn.predict(x_test)
        error_rate.append(np.mean(y_pred != y_test))
    kloc = error_rate.index(min(error_rate))
    print("Lowest error is %s occurs at k=%s." % (error_rate[kloc], K[kloc]))
```

```
    plt.plot(K, error_rate, color='blue', linestyle='dashed', marker='o',
             markerfacecolor='red', markersize=10)
    plt.title('Error Rate vs. K Value')
    plt.xlabel('K')
    plt.ylabel('Error Rate')
    plt.show()
    return K[kloc]
```

In [15]:

```
k=knn_error(21,x_train,y_train,x_test,y_test)
```

```
Lowest error is 0.021030494216614 09 occurs at k=4.
```



In [16]:

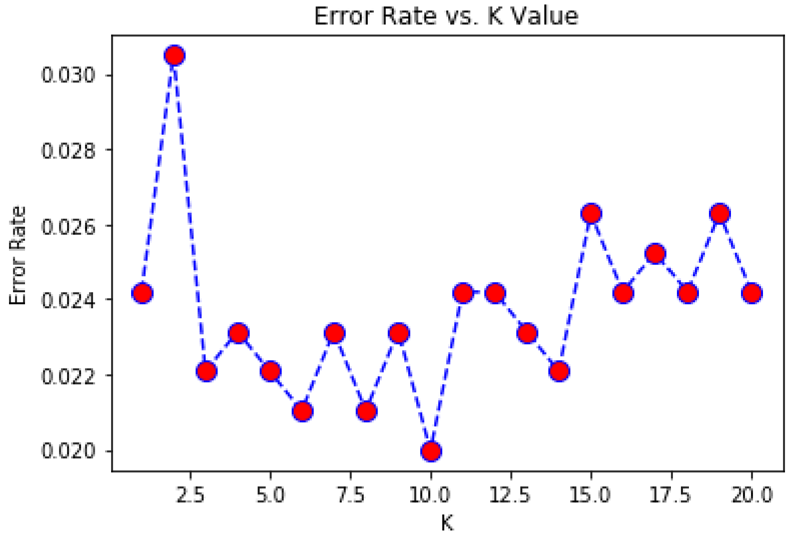
```
model = neighbors.KNeighborsClassifier(n_neighbors = k)
classify(model,x_train,y_train,x_test,y_test)
```

	precision	recall	f1-score	support
female	0.9772	0.9812	0.9792	480
male	0.9808	0.9766	0.9787	471
micro avg	0.9790	0.9790	0.9790	951
macro avg	0.9790	0.9789	0.9790	951
weighted avg	0.9790	0.9790	0.9790	951

In [17]:

```
k=knn_error(21,x_train3,y_train3,x_test3,y_test3)
```

```
Lowest error is 0.019978969505783387 occurs at k=10.
```



In [18]:

```
model = neighbors.KNeighborsClassifier(n_neighbors = k)
classify(model,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

      female     0.9789    0.9688    0.9738       480
        male     0.9685    0.9788    0.9736       471

   micro avg     0.9737    0.9737    0.9737       951
   macro avg     0.9737    0.9738    0.9737       951
weighted avg     0.9738    0.9737    0.9737       951
```

Naive Bayes

Using naive_bayes.GaussianNB() to build the model.

In [19]:

```
model=naive_bayes.GaussianNB()
classify(model,x_train,y_train,x_test,y_test)
```

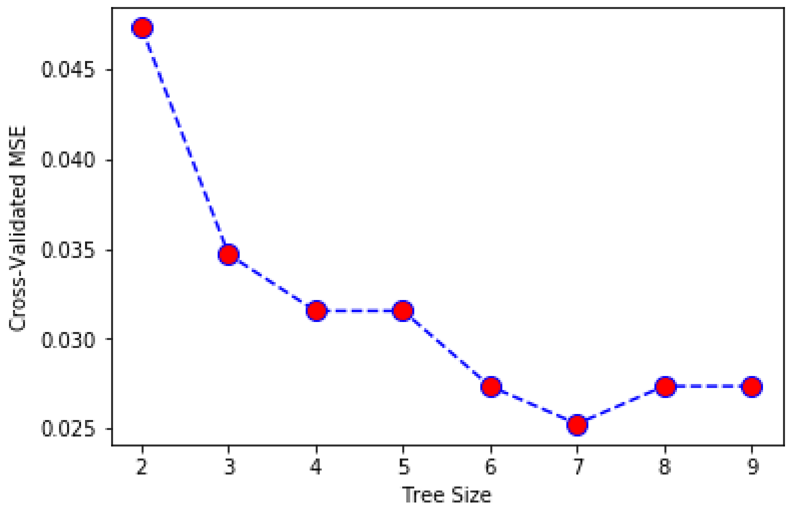
```
              precision    recall  f1-score   support

      female     0.9234    0.9042    0.9137       480
        male     0.9044    0.9236    0.9139       471

   micro avg     0.9138    0.9138    0.9138       951
   macro avg     0.9139    0.9139    0.9138       951
weighted avg     0.9140    0.9138    0.9138       951
```

In [20]:
```
model=naive_bayes.GaussianNB()
classify(model,x_train3,y_train3,x_test3,y_test3)
```
```
              precision    recall  f1-score   support

      female     0.9788    0.9625    0.9706       480
        male     0.9624    0.9788    0.9705       471

   micro avg     0.9706    0.9706    0.9706       951
   macro avg     0.9706    0.9706    0.9706       951
weighted avg     0.9707    0.9706    0.9706       951
```

Decision Tree

Using tree.DecisionTreeClassifier() to build the model.

In [21]:
```
#Find the best parameter to prune the tree
def dt_error(n,x_train,y_train,x_test,y_test):
    nodes = range(2, n)
    error_rate = []
    for k in nodes:
        model = tree.DecisionTreeClassifier(max_leaf_nodes=k)
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)
        error_rate.append(np.mean(y_pred != y_test))
    kloc = error_rate.index(min(error_rate))
    print("Lowest error is %s occurs at n=%s." % (error_rate[kloc], nodes[kloc]
))
    plt.plot(nodes, error_rate, color='blue', linestyle='dashed', marker='o',
             markerfacecolor='red', markersize=10)
    plt.xlabel('Tree Size')
    plt.ylabel('Cross-Validated MSE')
    plt.show()
    return nodes[kloc]
```
In [22]:
```
n=dt_error(10,x_train,y_train,x_test,y_test)
```
```
Lowest error is 0.025236593059936908 occurs at n=7.
```

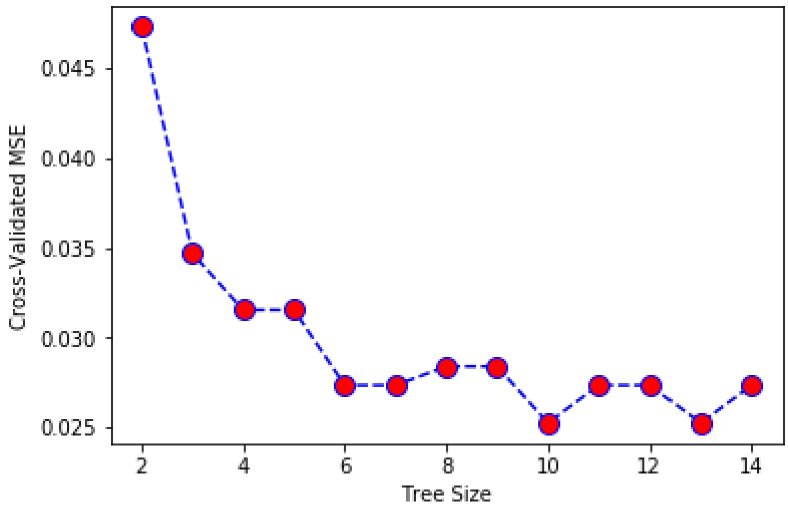


In [23]:

```
#prune tree
pruned_tree = tree.DecisionTreeClassifier(criterion = 'gini', max_leaf_nodes =
n)
classify(pruned_tree,x_train,y_train,x_test,y_test)
              precision    recall  f1-score   support

      female     0.9730    0.9771    0.9751       480
        male     0.9765    0.9724    0.9745       471

   micro avg     0.9748    0.9748    0.9748       951
   macro avg     0.9748    0.9747    0.9748       951
weighted avg     0.9748    0.9748    0.9748       951
```

In [24]:

```
n=dt_error(15,x_train3,y_train3,x_test3,y_test3)
Lowest error is 0.025236593059936908 occurs at n=10.
```



In [25]:

```
#prune tree
pruned_tree = tree.DecisionTreeClassifier(criterion = 'gini', max_leaf_nodes =
n)
classify(pruned_tree,x_train3,y_train3,x_test3,y_test3)
              precision    recall  f1-score   support

      female     0.9711    0.9792    0.9751       480
        male     0.9786    0.9703    0.9744       471

   micro avg     0.9748    0.9748    0.9748       951
   macro avg     0.9748    0.9747    0.9748       951
weighted avg     0.9748    0.9748    0.9748       951
```

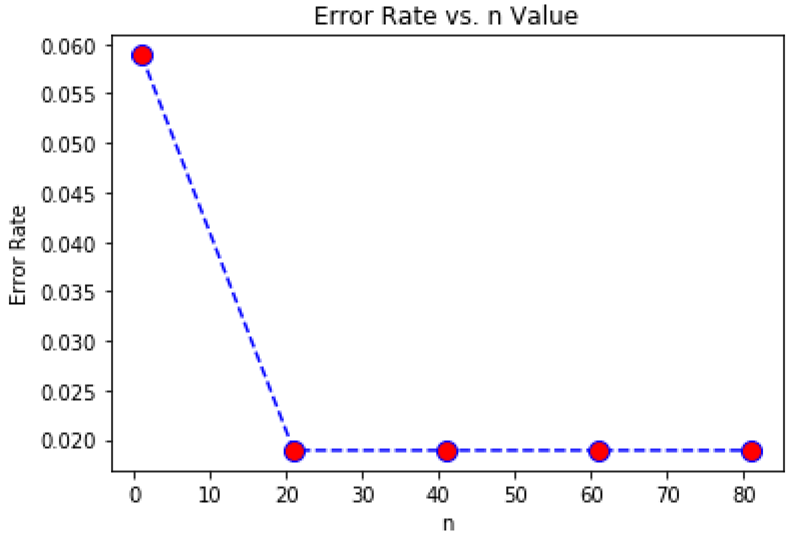
Random Forest

Using ensemble.RandomForestClassifier() to build the model.

In [26]:
```
def rf_error(n,x_train,y_train,x_test,y_test):
    error_rate = []
    e=range(1,n,20)
    for i in e:
        model = ensemble.RandomForestClassifier(n_estimators = i)
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)
        error_rate.append(np.mean(y_pred != y_test))
    nloc = error_rate.index(min(error_rate))
    print("Lowest error is %s occurs at n=%s." % (error_rate[nloc], e[nloc]))

    plt.plot(e, error_rate, color='blue', linestyle='dashed', marker='o',
             markerfacecolor='red', markersize=10)
    plt.title('Error Rate vs. n Value')
    plt.xlabel('n')
    plt.ylabel('Error Rate')
    plt.show()
    return e[nloc]
```

In [27]:
```
e=rf_error(100,x_train,y_train,x_test,y_test)
```

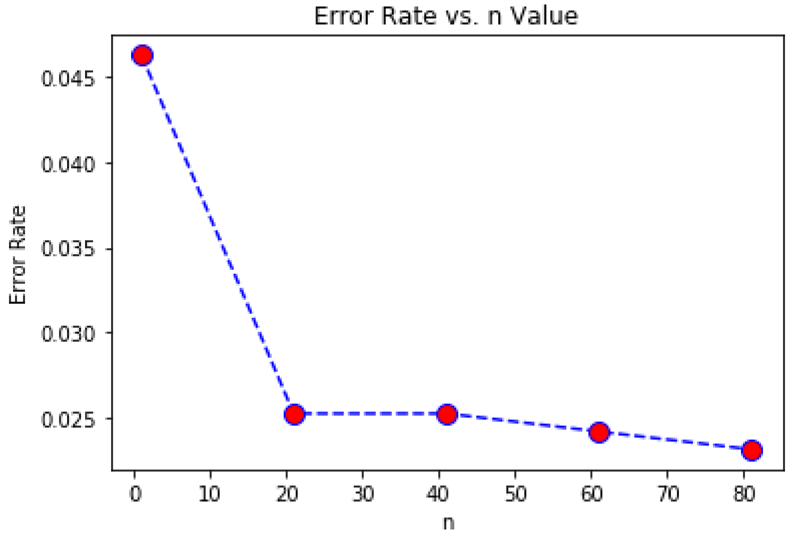
```
Lowest error is 0.018927444794952682 occurs at n=21.
```



In [28]:
```
model=ensemble.RandomForestClassifier(n_estimators = e)
classify(model,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

      female     0.9752    0.9833    0.9793       480
        male     0.9829    0.9745    0.9787       471

   micro avg     0.9790    0.9790    0.9790       951
   macro avg     0.9790    0.9789    0.9790       951
weighted avg     0.9790    0.9790    0.9790       951
```

In [29]:
```
e=rf_error(100,x_train3,y_train3,x_test3,y_test3)
Lowest error is 0.023133543638275498 occurs at n=81.
```



In [30]:
```
model=ensemble.RandomForestClassifier(n_estimators = e)
classify(model,x_train3,y_train3,x_test3,y_test3)
              precision    recall  f1-score   support

      female     0.9731    0.9792    0.9761       480
        male     0.9786    0.9724    0.9755       471

   micro avg     0.9758    0.9758    0.9758       951
   macro avg     0.9759    0.9758    0.9758       951
weighted avg     0.9758    0.9758    0.9758       951
```

XgBoost

Using xgboost.XGBClassifier() to build the model.

In [31]:
```
model = xgboost.XGBClassifier()
classify(model,x_train,y_train,x_test,y_test)
              precision    recall  f1-score   support

      female     0.9793    0.9854    0.9823       480
        male     0.9850    0.9788    0.9819       471

   micro avg     0.9821    0.9821    0.9821       951
   macro avg     0.9822    0.9821    0.9821       951
weighted avg     0.9821    0.9821    0.9821       951
```

In [32]:
```
model = xgboost.XGBClassifier()
classify(model,x_train3,y_train3,x_test3,y_test3)
```
```
              precision    recall  f1-score   support

      female     0.9731    0.9792    0.9761       480
        male     0.9786    0.9724    0.9755       471

   micro avg     0.9758    0.9758    0.9758       951
   macro avg     0.9759    0.9758    0.9758       951
weighted avg     0.9758    0.9758    0.9758       951
```

linkcode

Support Vector Machine¶

Using svm.SVC() to build the model.

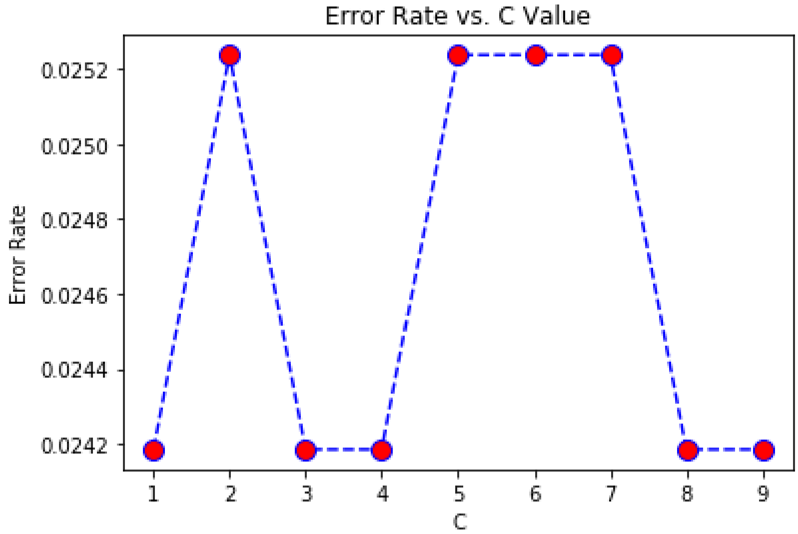
In [33]:
```
def svm_kernel(x_train,y_train,x_test,y_test):
    rate=[]
    kernel=['rbf','poly','linear']
    for i in kernel:
        model=svm.SVC(kernel=i).fit(x_train,y_train)
        y_pred=model.predict(x_train)
        print(i, ' in-sample accuracy in SVM: ', accuracy_score(y_train,y_pred)
)
        y_pred=model.predict(x_test)
        print(i, ' out-of-sample accuracy in SVM: ', accuracy_score(y_test,y_pr
ed))
        rate.append(accuracy_score(y_test,y_pred))
    nloc = rate.index(max(rate))
    print("Highest accuracy is %s occurs at %s kernel." % (rate[nloc], kernel[n
loc]))
    return kernel[nloc]
```
In [34]:
```
def svm_error(k,C,x_train,y_train,x_test,y_test):
    error_rate = []
    C=range(1,C)
    for i in C:
        model=svm.SVC(kernel=k,C=i).fit(x_train,y_train)
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)
        error_rate.append(np.mean(y_pred != y_test))
    cloc = error_rate.index(min(error_rate))
    print("Lowest error is %s occurs at C=%s." % (error_rate[cloc], C[cloc]))

    plt.plot(C, error_rate, color='blue', linestyle='dashed', marker='o',
             markerfacecolor='red', markersize=10)
    plt.title('Error Rate vs. C Value')
    plt.xlabel('C')
    plt.ylabel('Error Rate')
    plt.show()
    return C[cloc]
```

In [35]:

```
k=svm_kernel(x_train,y_train,x_test,y_test)
rbf  in-sample accuracy in SVM:  0.9657194406856112
rbf  out-of-sample accuracy in SVM:  0.9737118822292324
poly  in-sample accuracy in SVM:  0.8700947225981055
poly  out-of-sample accuracy in SVM:  0.8769716088328076
linear  in-sample accuracy in SVM:  0.972936400541272
linear  out-of-sample accuracy in SVM:  0.9758149316508938
Highest accuracy is 0.9758149316508938 occurs at linear kernel.
```

In [36]:

```
c=svm_error(k,10,x_train,y_train,x_test,y_test)
Lowest error is 0.024185068349106203 occurs at C=1.
```

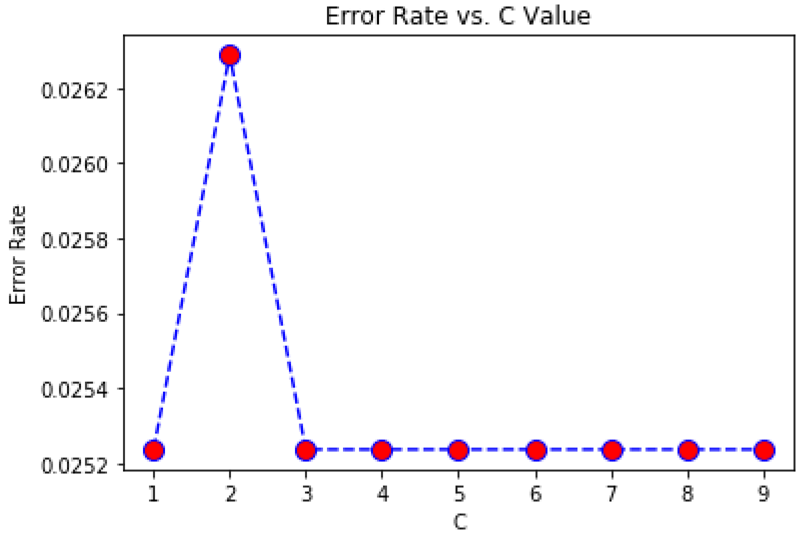


In [37]:

```
model=svm.SVC(kernel=k,C=c)
classify(model,x_train,y_train,x_test,y_test)
              precision    recall  f1-score   support

      female     0.9751    0.9771    0.9761       480
        male     0.9766    0.9745    0.9756       471

   micro avg     0.9758    0.9758    0.9758       951
   macro avg     0.9758    0.9758    0.9758       951
weighted avg     0.9758    0.9758    0.9758       951
```

In [38]:

```
k=svm_kernel(x_train3,y_train3,x_test3,y_test3)
rbf  in-sample accuracy in SVM:  0.9661705006765899
rbf  out-of-sample accuracy in SVM:  0.9747634069400631
poly  in-sample accuracy in SVM:  0.9404600811907984
poly  out-of-sample accuracy in SVM:  0.9484752891692955
linear  in-sample accuracy in SVM:  0.963915200721696
linear  out-of-sample accuracy in SVM:  0.9726603575184016
Highest accuracy is 0.9747634069400631 occurs at rbf kernel.
```

In [39]:
```
c=svm_error(k,10,x_train3,y_train3,x_test3,y_test3)
```
```
Lowest error is 0.025236593059936908 occurs at C=1.
```


In [40]:
```
model=svm.SVC(kernel=k,C=c)
classify(model,x_train3,y_train3,x_test3,y_test3)
```
```
              precision    recall  f1-score   support

      female     0.9810    0.9688    0.9748       480
        male     0.9686    0.9809    0.9747       471

   micro avg     0.9748    0.9748    0.9748       951
   macro avg     0.9748    0.9748    0.9748       951
weighted avg     0.9748    0.9748    0.9748       951
```

Neural Network

Using neural_network.MLPClassifier to build the model.

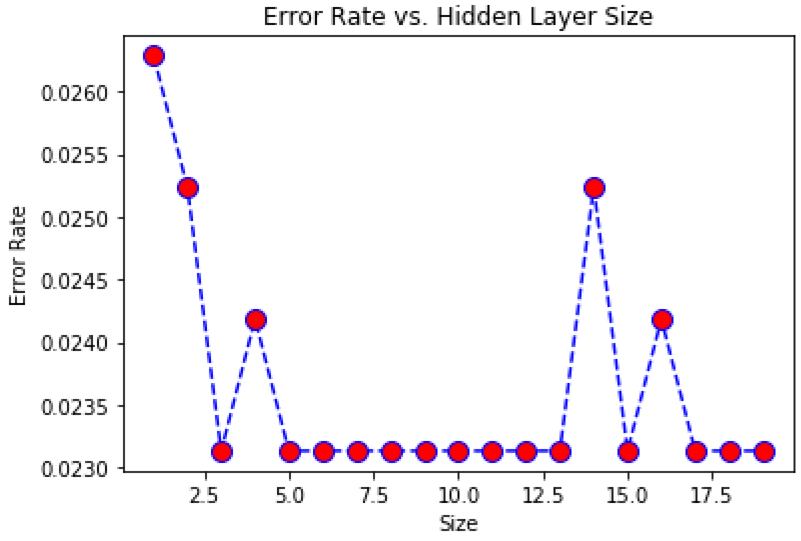
In [41]:
```
def nn_error(n,x_train,y_train,x_test,y_test):
    error_rate = []
    hidden_layer=range(1,n)
    for i in hidden_layer:
        model = neural_network.MLPClassifier(solver='adam', alpha=1e-5,
                                              hidden_layer_sizes=i,
                                              activation='logistic',random_state=17,
                                              max_iter=2000)
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)
        error_rate.append(np.mean(y_pred != y_test))
    kloc = error_rate.index(min(error_rate))
    print("Lowest error is %s occurs at C=%s." % (error_rate[kloc], hidden_laye
r[kloc]))
```

```
    plt.plot(hidden_layer, error_rate, color='blue', linestyle='dashed', marker
='o',
             markerfacecolor='red', markersize=10)
    plt.title('Error Rate vs. Hidden Layer Size')
    plt.xlabel('Size')
    plt.ylabel('Error Rate')
    plt.show()
    return hidden_layer[kloc]
```

In [42]:

```
h=nn_error(20,x_train,y_train,x_test,y_test)
```

```
Lowest error is 0.023133543638275498 occurs at C=3.
```



In [43]:

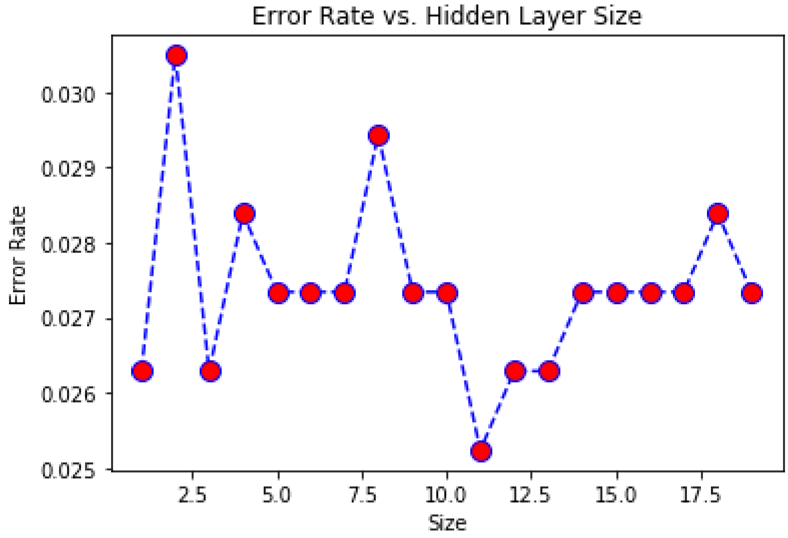
```
model = neural_network.MLPClassifier(solver='adam', alpha=1e-5,
                                     hidden_layer_sizes=h,
                                     activation='logistic',random_state=17,
                                     max_iter=2000)
classify(model,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

      female     0.9771    0.9771    0.9771       480
        male     0.9766    0.9766    0.9766       471

   micro avg     0.9769    0.9769    0.9769       951
   macro avg     0.9769    0.9769    0.9769       951
weighted avg     0.9769    0.9769    0.9769       951
```

In [44]:

```
h=nn_error(20,x_train3,y_train3,x_test3,y_test3)
```

```
Lowest error is 0.025236593059936908 occurs at C=11.
```



In [45]:

```
model = neural_network.MLPClassifier(solver='adam', alpha=1e-5,
                                     hidden_layer_sizes=h,
                                     activation='logistic',random_state=17,
                                     max_iter=2000)
classify(model,x_train3,y_train3,x_test3,y_test3)
```

```
              precision    recall  f1-score   support

      female     0.9730    0.9771    0.9751       480
        male     0.9765    0.9724    0.9745       471

   micro avg     0.9748    0.9748    0.9748       951
   macro avg     0.9748    0.9747    0.9748       951
weighted avg     0.9748    0.9748    0.9748       951
```

RESULTS

Seven machine learning algorithms were employed for model building:

K-Nearest Neighbors (KNN): KNN achieved an accuracy of 97.9% using the full set of features and 97.4% using only the three most significant features (meanfun, IQR, Q25).

Naive Bayes: Gaussian Naive Bayes achieved an accuracy of 91.4% with the full feature set and 97.1% with the three most significant features.

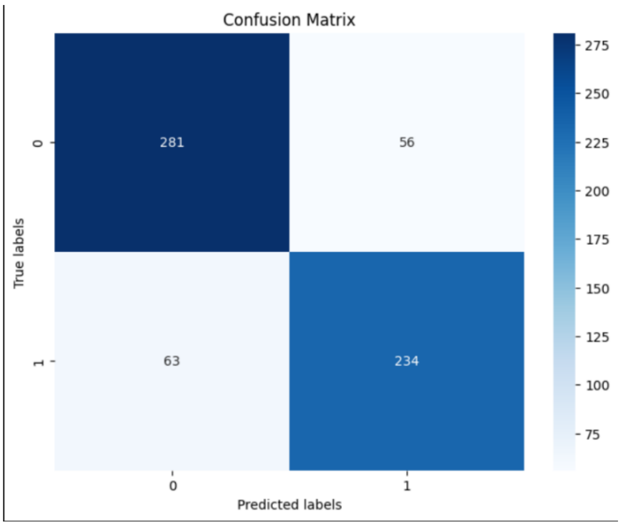
Decision Tree: Pruned decision trees attained an accuracy of 97.5% with both the full feature set and the selected three features.

Random Forest: Random Forest yielded an accuracy of 97.9% using the full set of features and 97.6% using the selected features.

XGBoost: XGBoost achieved an accuracy of 98.2% with both the full feature set and the selected features.

Support Vector Machine (SVM): SVM with a linear kernel attained the highest accuracy of 97.6% with the full feature set and 97.5% with the selected features.

Neural Network: The MLPClassifier achieved an accuracy of 97.5% with the full feature set and 97.5% .**Confusion Matrix**



CONCLUSION

Overall, all the models performed well in classifying the gender based on voice features. XGBoost exhibited the highest accuracy among the models tested, closely followed by KNN and SVM with a linear kernel. The selected features subset also provided competitive performance, indicating the significance of those features in gender classification. Further fine-tuning and optimization of the models could potentially improve performance, but the current results demonstrate the effectiveness of machine learning algorithms in voice gender classification

REFERENCES

https://www.youtube.com/watch?v=JxgmHe2NyeY

https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

https://www.cs.cmu.edu/%7Etom/10701_sp11/lectures.shtml

CENTRAL UNIVERSITY OF KARNATAKA

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HEAD OF DEPARTMENT: DR .PARMESHA sir.

SUBJECT :

CO-ORDINATOR : Dr. Nagaraj.Y