Feature Engineering

o Machine Learning

- a. Supervised Machine Learning -- A. Regression
  - Linear Regression, Polynomial Regression, Lasso, Ridge etc.

-- B. Classification - Logistic regression, Decision Tree, Random Forest , SVM, Naive Bayes

- Feature Enginnering
- a. Feature Selection
  - Dimensionality Reduction
- a. Feature Reduction
  - a. Supervised ML : LDA : Linear Discriminant Analysis
  - b. Un Supervised ML : PCA : Principal Component Analysis

-- Hyperparameter Tuning 1 -- Balance Data

- a. UnSupupervised Machine Learning : 2-3

-- Computer Vision 2 -- Natural Language Processing 2-3 -- Recommendation Systems -- Time Series Analysis

```
# Import Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('voice-classification.csv')

df.shape
```

```
(3168, 21)
```

```
df.columns
```

```
Index(['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew',
'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun',
'maxfun',
```

```
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx', 'label'],
      dtype='object')
```

```
df['label'].unique()
```

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

```
# Imbalance in Data
```

```
df['label'].value_counts()
```

```
label
male      1584
female    1584
Name: count, dtype: int64
```

```
# info()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   meanfreq  3168 non-null   float64
 1   sd        3168 non-null   float64
 2   median    3168 non-null   float64
 3   Q25       3168 non-null   float64
 4   Q75       3168 non-null   float64
 5   IQR       3168 non-null   float64
 6   skew      3168 non-null   float64
 7   kurt      3168 non-null   float64
 8   sp.ent    3168 non-null   float64
 9   sfm       3168 non-null   float64
 10  mode      3168 non-null   float64
 11  centroid  3168 non-null   float64
 12  meanfun   3168 non-null   float64
 13  minfun    3168 non-null   float64
 14  maxfun    3168 non-null   float64
 15  meandom   3168 non-null   float64
 16  mindom    3168 non-null   float64
 17  maxdom    3168 non-null   float64
 18  dfrange   3168 non-null   float64
 19  modindx   3168 non-null   float64
 20  label     3168 non-null   object 
dtypes: float64(20), object(1)
memory usage: 519.9+ KB
```

```
# data types

df.dtypes
```

```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label        object
dtype: object
```

```
df.describe().round(2)
```

```
       meanfreq       sd    median      Q25      Q75      IQR     skew
\
count   3168.00  3168.00  3168.00  3168.00  3168.00  3168.00  3168.00

mean       0.18     0.06     0.19     0.14     0.22     0.08     3.14

std        0.03     0.02     0.04     0.05     0.02     0.04     4.24

min        0.04     0.02     0.01     0.00     0.04     0.01     0.14

25%        0.16     0.04     0.17     0.11     0.21     0.04     1.65

50%        0.18     0.06     0.19     0.14     0.23     0.09     2.20

75%        0.20     0.07     0.21     0.18     0.24     0.11     2.93

max        0.25     0.12     0.26     0.25     0.27     0.25    34.73


          kurt   sp.ent      sfm     mode  centroid  meanfun   minfun
\
count  3168.00  3168.00  3168.00  3168.00   3168.00  3168.00  3168.00
```

```
mean     36.57     0.90     0.41     0.17      0.18     0.14     0.04

std     134.93     0.04     0.18     0.08      0.03     0.03     0.02

min       2.07     0.74     0.04     0.00      0.04     0.06     0.01

25%       5.67     0.86     0.26     0.12      0.16     0.12     0.02

50%       8.32     0.90     0.40     0.19      0.18     0.14     0.05

75%      13.65     0.93     0.53     0.22      0.20     0.17     0.05

max    1309.61     0.98     0.84     0.28      0.25     0.24     0.20
```

```
        maxfun  meandom   mindom   maxdom  dfrange  modindx
count  3168.00  3168.00  3168.00  3168.00  3168.00  3168.00
mean      0.26     0.83     0.05     5.05     4.99     0.17
std       0.03     0.53     0.06     3.52     3.52     0.12
min       0.10     0.01     0.00     0.01     0.00     0.00
25%       0.25     0.42     0.01     2.07     2.04     0.10
50%       0.27     0.77     0.02     4.99     4.95     0.14
75%       0.28     1.18     0.07     7.01     6.99     0.21
max       0.28     2.96     0.46    21.87    21.84     0.93
```

```
# Convert label into Numbers. LabelEncoder()
```

```
df['label'].unique()
```

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

```
from sklearn.preprocessing import LabelEncoder
```

```
LE = LabelEncoder()
```

```
df['label'] = LE.fit_transform( df['label'])
```

```
df['label']             # 1 : Male         0 : Female
```

```
0       1
1       1
2       1
3       1
4       1
       ..
3163    0
3164    0
3165    0
3166    0
3167    0
Name: label, Length: 3168, dtype: int32
```

```
df['label']
```

```
0       1
1       1
2       1
3       1
4       1
       ..
3163    0
3164    0
3165    0
3166    0
3167    0
Name: label, Length: 3168, dtype: int32
```

```
# Correlation
```

```
df.corr().round(2)
```

```
          meanfreq    sd  median   Q25   Q75   IQR  skew  kurt  sp.ent
sfm  \
meanfreq      1.00 -0.74    0.93  0.91  0.74 -0.63 -0.32 -0.32   -0.60
-0.78
sd           -0.74  1.00   -0.56 -0.85 -0.16  0.87  0.31  0.35    0.72
0.84
median        0.93 -0.56    1.00  0.77  0.73 -0.48 -0.26 -0.24   -0.50
-0.66
Q25           0.91 -0.85    0.77  1.00  0.48 -0.87 -0.32 -0.35   -0.65
-0.77
Q75           0.74 -0.16    0.73  0.48  1.00  0.01 -0.21 -0.15   -0.17
-0.38
IQR          -0.63  0.87   -0.48 -0.87  0.01  1.00  0.25  0.32    0.64
0.66
skew         -0.32  0.31   -0.26 -0.32 -0.21  0.25  1.00  0.98   -0.20
0.08
kurt         -0.32  0.35   -0.24 -0.35 -0.15  0.32  0.98  1.00   -0.13
0.11
sp.ent       -0.60  0.72   -0.50 -0.65 -0.17  0.64 -0.20 -0.13    1.00
0.87
sfm          -0.78  0.84   -0.66 -0.77 -0.38  0.66  0.08  0.11    0.87
1.00
mode          0.69 -0.53    0.68  0.59  0.49 -0.40 -0.43 -0.41   -0.33
-0.49
centroid      1.00 -0.74    0.93  0.91  0.74 -0.63 -0.32 -0.32   -0.60
-0.78
meanfun       0.46 -0.47    0.41  0.55  0.16 -0.53 -0.17 -0.19   -0.51
-0.42
minfun        0.38 -0.35    0.34  0.32  0.26 -0.22 -0.22 -0.20   -0.31
-0.36
maxfun        0.27 -0.13    0.25  0.20  0.29 -0.07 -0.08 -0.05   -0.12
```

```
-0.19
meandom       0.54 -0.48    0.46  0.47  0.36 -0.33 -0.34 -0.30   -0.29
-0.43
mindom        0.23 -0.36    0.19  0.30 -0.02 -0.36 -0.06 -0.10   -0.29
-0.29
maxdom        0.52 -0.48    0.44  0.46  0.34 -0.34 -0.31 -0.27   -0.32
-0.44
dfrange       0.52 -0.48    0.44  0.45  0.34 -0.33 -0.30 -0.27   -0.32
-0.43
modindx      -0.22  0.12   -0.21 -0.14 -0.22  0.04 -0.17 -0.21    0.20
0.21
label        -0.34  0.48   -0.28 -0.51  0.07  0.62  0.04  0.09    0.49
0.36

              ...  centroid  meanfun  minfun  maxfun  meandom  mindom
maxdom  \
meanfreq      ...      1.00     0.46    0.38    0.27     0.54    0.23
0.52
sd            ...     -0.74    -0.47   -0.35   -0.13    -0.48   -0.36   -
0.48
median        ...      0.93     0.41    0.34    0.25     0.46    0.19
0.44
Q25           ...      0.91     0.55    0.32    0.20     0.47    0.30
0.46
Q75           ...      0.74     0.16    0.26    0.29     0.36   -0.02
0.34
IQR           ...     -0.63    -0.53   -0.22   -0.07    -0.33   -0.36   -
0.34
skew          ...     -0.32    -0.17   -0.22   -0.08    -0.34   -0.06   -
0.31
kurt          ...     -0.32    -0.19   -0.20   -0.05    -0.30   -0.10   -
0.27
sp.ent        ...     -0.60    -0.51   -0.31   -0.12    -0.29   -0.29   -
0.32
sfm           ...     -0.78    -0.42   -0.36   -0.19    -0.43   -0.29   -
0.44
mode          ...      0.69     0.32    0.39    0.17     0.49    0.20
0.48
centroid      ...      1.00     0.46    0.38    0.27     0.54    0.23
0.52
meanfun       ...      0.46     1.00    0.34    0.31     0.27    0.16
0.28
minfun        ...      0.38     0.34    1.00    0.21     0.38    0.08
0.32
maxfun        ...      0.27     0.31    0.21    1.00     0.34   -0.24
0.36
meandom       ...      0.54     0.27    0.38    0.34     1.00    0.10
0.81
mindom        ...      0.23     0.16    0.08   -0.24     0.10    1.00
```

```
0.03
maxdom    ...       0.52     0.28    0.32    0.36     0.81    0.03
1.00
dfrange   ...       0.52     0.28    0.32    0.36     0.81    0.01
1.00
modindx   ...      -0.22    -0.05    0.00   -0.36    -0.18    0.20   -
0.43
label     ...      -0.34    -0.83   -0.14   -0.17    -0.19   -0.19   -
0.20

          dfrange  modindx  label
meanfreq     0.52    -0.22  -0.34
sd          -0.48     0.12   0.48
median       0.44    -0.21  -0.28
Q25          0.45    -0.14  -0.51
Q75          0.34    -0.22   0.07
IQR         -0.33     0.04   0.62
skew        -0.30    -0.17   0.04
kurt        -0.27    -0.21   0.09
sp.ent      -0.32     0.20   0.49
sfm         -0.43     0.21   0.36
mode         0.47    -0.18  -0.17
centroid     0.52    -0.22  -0.34
meanfun      0.28    -0.05  -0.83
minfun       0.32     0.00  -0.14
maxfun       0.36    -0.36  -0.17
meandom      0.81    -0.18  -0.19
mindom       0.01     0.20  -0.19
maxdom       1.00    -0.43  -0.20
dfrange      1.00    -0.43  -0.19
modindx     -0.43     1.00   0.03
label       -0.19     0.03   1.00

[21 rows x 21 columns]
```

```
# Decide I/P and O/P

X = df.drop('label', axis = 1 )
Y = df['label']

X.shape
```

```
(3168, 20)
```

```
# Total 20 Features

# Split Data

from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=.20, random_state=12)

X_train.shape, X_test.shape, Y_train.shape, Y_test.shape

((2534, 20), (634, 20), (2534,), (634,))

# Apply Logistic Regression , Decision Tree, Random Forest, SVM, Naive
Bayes  => Evaluate
```

1. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

LR = LogisticRegression()

# Train Model
LR.fit(X_train, Y_train)

LogisticRegression()

# Predict Outcomes
LR_pred = LR.predict(X_test)

# Evaluate

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

confusion_matrix(Y_test, LR_pred)

array([[269,  62],
       [ 11, 292]], dtype=int64)

accuracy_score(Y_test, LR_pred).round(2)

0.88

print(classification_report(Y_test, LR_pred))

              precision    recall  f1-score   support

           0       0.96      0.81      0.88       331
           1       0.82      0.96      0.89       303

    accuracy                           0.88       634
   macro avg       0.89      0.89      0.88       634
weighted avg       0.90      0.88      0.88       634
```

2. Decision Tree

```
DT = DecisionTreeClassifier()

# Train Model
DT.fit(X_train, Y_train)

DecisionTreeClassifier()

# Predict Outcome

DT_pred = DT.predict(X_test)

# Evaluate

accuracy_score(Y_test, DT_pred)

0.973186119873817
```

3. Random Forest

```
RF = RandomForestClassifier()

# Train Model

RF.fit(X_train, Y_train)

RandomForestClassifier()

# Predict Outcom e
RF_pred = RF.predict(X_test)

accuracy_score(Y_test, RF_pred).round(2)

0.98
```

3. Support Vector Classifier

```
SV = SVC()

# Train Model

SV.fit(X_train, Y_train)

SVC()

# Predict Outcome

SV_pred = SV.predict(X_test)

# Evaluate

accuracy_score(Y_test , SV_pred)

0.6403785488958991
```

4. Naive Bayes

```
NV = GaussianNB()

# Train model

NV.fit(X_train, Y_train)

GaussianNB()

# Predict Outcomes

NV_pred = NV.predict(X_test)

# Evaluate

accuracy_score(Y_test, NV_pred)

0.8706624605678234
```

Dimensionality Reduction / Feature Reduction / Feature Engineering :

1. Linear Discriminant Analysis

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

LDA = LinearDiscriminantAnalysis()

X_train.shape, X_test.shape

((2534, 20), (634, 20))

# Output Column / Feature : Categorical

# Classes : 'Male' , "Female"

# Number of components (<= min(n_classes - 1, n_features)) for
dimensionality reduction.

# No of Features = min(2-1, 20)    => min(1,20)  =>  1

LDA.fit(X_train, Y_train)

LinearDiscriminantAnalysis()

X_train_lda = LDA.transform(X_train)
X_test_lda  = LDA.transform(X_test)

X_train_lda.shape, X_test_lda.shape

((2534, 1), (634, 1))

# 1. Logistic Regression
```

```
LR.fit(X_train_lda, Y_train)
```

```
LogisticRegression()
```

```
accuracy_score(Y_test, LR.predict(X_test_lda))
```

```
0.9700315457413249
```

```
# 2. Decsision Tree
```

```
DT.fit(X_train_lda , Y_train)
```

```
DecisionTreeClassifier()
```

```
accuracy_score(Y_test, DT.predict(X_test_lda))
```

```
0.944794952681388
```

```
# 3. Random Forest
```

```
RF.fit(X_train_lda, Y_train)
```

```
RandomForestClassifier()
```

```
accuracy_score(Y_test, RF.predict(X_test_lda)).round(2)
```

```
0.94
```

```
# 4.Support Vector Classifier
```

```
SV.fit(X_train_lda, Y_train)
```

```
SVC()
```

```
accuracy_score(Y_test, SV.predict(X_test_lda)).round(2)
```

```
0.97
```

```
# 5. Naive Bayes
```

```
accuracy_score(Y_test, NV.fit(X_train_lda,
Y_train).predict(X_test_lda))
```

```
0.9700315457413249
```

```
                                  accuracy score         Dimensionality
Reduction
                                                                    LDA +
```

1. Logistic Regression 88% 97%
2. Decision Tree 97% 94%
3. Random Forest 98% 94%
4. Support Vector Classifier 64% 97%
5. Naive Bayes 87% 97%