In [1]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
```

In [3]:
```python
df = pd.read_csv('Resume_dataset.csv')
```

In [4]:
```python
df.head()
```

Out[4]:

	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...

In [5]:
```python
df.isnull().sum()
```

Out[5]:
```
Category    0
Resume      0
dtype: int64
```

In [6]:
```python
print("Displaying the distinct categories of resume -\n")
print(df['Category'].unique())
```

```
Displaying the distinct categories of resume -

['Data Science' 'HR' 'Advocate' 'Arts' 'Web Designing'
 'Mechanical Engineer' 'Sales' 'Health and fitness' 'Civil Engineer'
 'Java Developer' 'Business Analyst' 'SAP Developer' 'Automation Testing'
 'Electrical Engineering' 'Operations Manager' 'Python Developer'
 'DevOps Engineer' 'Network Security Engineer' 'PMO' 'Database' 'Hadoop'
 'ETL Developer' 'DotNet Developer' 'Blockchain' 'Testing']
```

In [7]:
```python
print("Displaying the number of records belonging to distinct categories of resume
print(df['Category'].value_counts())
```

```
Displaying the number of records belonging to distinct categories of resume -

Category
Java Developer               14
Database                     11
HR                           11
Data Science                 10
Advocate                     10
DotNet Developer              7
Hadoop                        7
DevOps Engineer               7
Automation Testing            7
Testing                       7
Civil Engineer                6
Business Analyst              6
SAP Developer                 6
Health and fitness            6
Python Developer              6
Arts                          6
Electrical Engineering        5
Sales                         5
Network Security Engineer     5
Mechanical Engineer           5
Web Designing                 5
ETL Developer                 5
Blockchain                    5
Operations Manager            4
PMO                           3
Name: count, dtype: int64
```

In [8]:

```
df['Category'].nunique()
```

Out[8]:

```
25
```

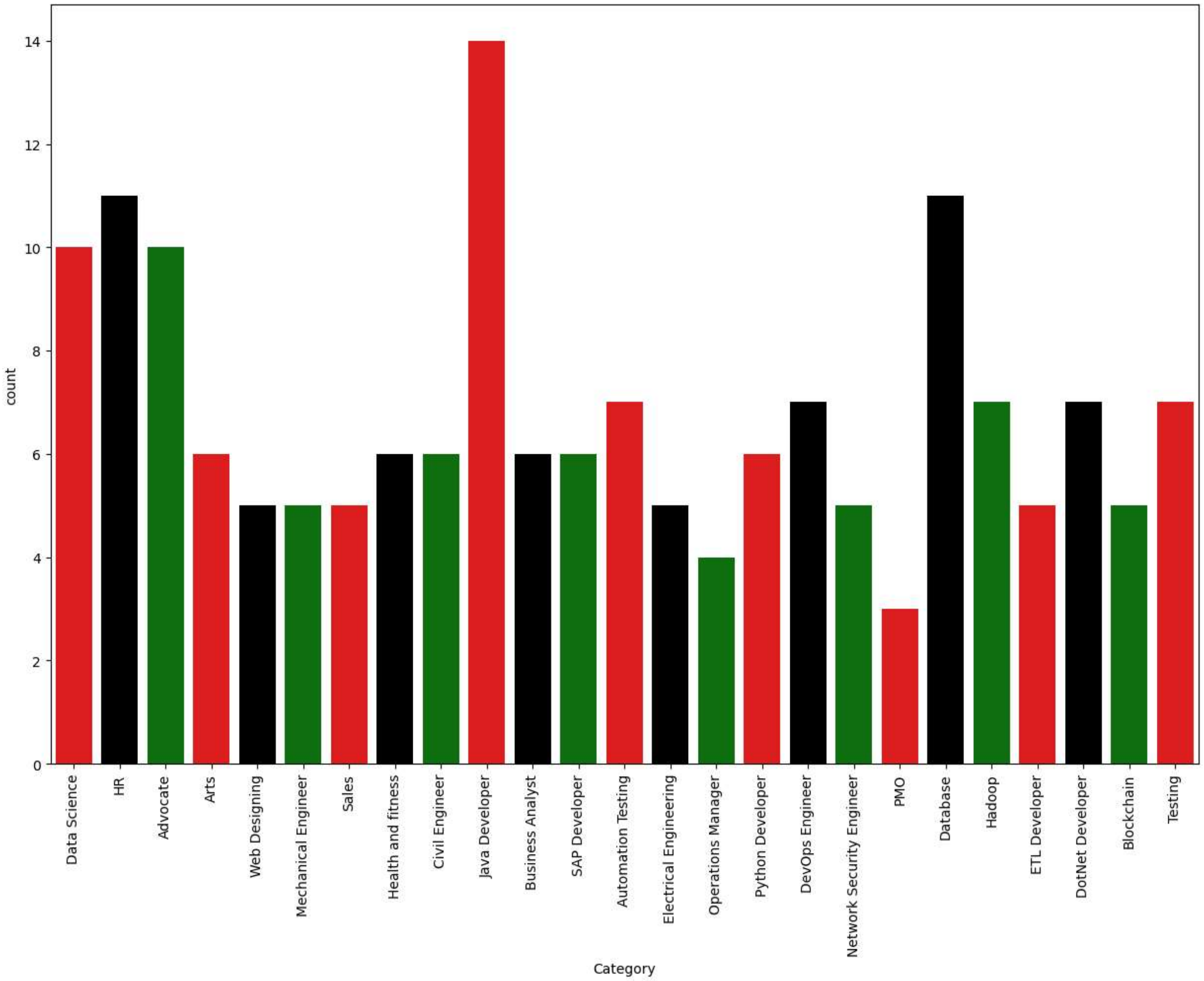
Visualization of category

In [9]:

```
plt.figure(figsize = (15, 10))
plt.xticks(rotation = 90)
sns.countplot(x = 'Category', data = df, palette = ['red', 'black', 'green'])
```

Out[9]:

```
<Axes: xlabel='Category', ylabel='count'>
```



Label Encoding

```
In [10]: le = LabelEncoder()
         df['Category'] = le.fit_transform(df['Category'])
         df.head()
```

Out[10]:

	Category	Resume
0	6	Skills * Programming Languages: Python (pandas...
1	6	Education Details \r\nMay 2013 to May 2017 B.E...
2	6	Areas of Interest Deep Learning, Control Syste...
3	6	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...
4	6	Education Details \r\n MCA YMCAUST, Faridab...

Text Preprocessing

```
In [11]: ps = PorterStemmer()
         stopwords = nltk.corpus.stopwords.words('english')
         def get_clean(x):
             x = re.sub('[^a-zA-Z]', ' ', x).lower().split()
```

```
    x = ' '.join([ps.stem(word) for word in x if word not in stopwords])
    return x
```

In [12]:
```
df['Resume'] = df['Resume'].apply(lambda x: get_clean(x))
```

In [13]:
```
df.head()
```

Out[13]:

	Category	Resume
0	6	skill program languag python panda numpi scipi...
1	6	educ detail may may b e uit rgpvdata scientist...
2	6	area interest deep learn control system design...
3	6	skill r python sap hana tableau sap hana sql s...
4	6	educ detail mca ymcaust faridabad haryanadata ...

TF-IDF Vectorizer

In [14]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features = 3000, ngram_range = (1, 3))
X = tfidf.fit_transform(df['Resume'])
y = df['Category']
```

In [15]:
```
print(tfidf.get_feature_names_out())
```

```
['abap' 'abil' 'abil work' ... 'year work' 'york' 'zone']
```

In [16]:
```
data_sample_1 = df[:2]
tfidf1 = TfidfVectorizer()

data_2_tfidf = tfidf1.fit_transform(data_sample_1['Resume'])
```

In [17]:
```
data_2_tfidf.shape
```

Out[17]:
```
(2, 346)
```

In [18]:
```
pip install --upgrade scikit-learn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in c:\users\ds_lab_18\appdata\roaming\py
thon\python312\site-packages (1.5.1)
Requirement already satisfied: numpy>=1.19.5 in c:\programdata\anaconda3\lib\site-pa
ckages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\programdata\anaconda3\lib\site-pac
kages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\programdata\anaconda3\lib\site-pa
ckages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\ds_lab_18\appdata\ro
aming\python\python312\site-packages (from scikit-learn) (3.5.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 24.1.2 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [19]:

```
df_tfidf = pd.DataFrame(data_2_tfidf.toarray(), columns = tfidf1.get_feature_names_
df_tfidf.head()
```

Out[19]:

	acceler	account	achiev	across	action	address	aditya	advanc	also
0	0.026709	0.026709	0.000000	0.080128	0.026709	0.026709	0.000000	0.026709	0.026709
1	0.000000	0.000000	0.069665	0.000000	0.000000	0.000000	0.069665	0.000000	0.000000

2 rows × 346 columns

In [20]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.20)
```

In [21]:

```
print('X_train shape :', X_train.shape)
print('y_train shape :', y_train.shape)
print('X_val shape :', X_val.shape)
print('y_val shape :', y_val.shape)
```

```
X_train shape : (135, 3000)
y_train shape : (135,)
X_val shape : (34, 3000)
y_val shape : (34,)
```

Logistic Regression

In [22]:

```
from sklearn.linear_model import LogisticRegression

log = LogisticRegression()
log.fit(X_train, y_train)
y_pred_log = log.predict(X_val)
```

In [23]:

```
print('Accuracy of Logistic Classifier: {:.2f}'.format(accuracy_score(y_val, y_pred
```

```
Accuracy of Logistic Classifier: 0.41
```

Grid Search

In [24]:

```
from sklearn.model_selection import GridSearchCV
```

In [25]:

```
log = LogisticRegression()
grid_values = {'penalty':['l1', 'l2'], 'C':[0.001, 0.01, 1, 5, 10, 25]}
grid_log_acc = GridSearchCV(log, param_grid = grid_values)
grid_log_acc.fit(X_train, y_train)
```

Out[25]:

```
▸ GridSearchCV
▸ best_estimator_: LogisticRegression
    ▸ LogisticRegression
```

In [26]:

```
#Predict values based on new parameters
y_pred_log_acc = grid_log_acc.predict(X_val)

#New Model Evaluation Metrics
print('Accuracy Score: ', accuracy_score(y_val, y_pred_log_acc))
```

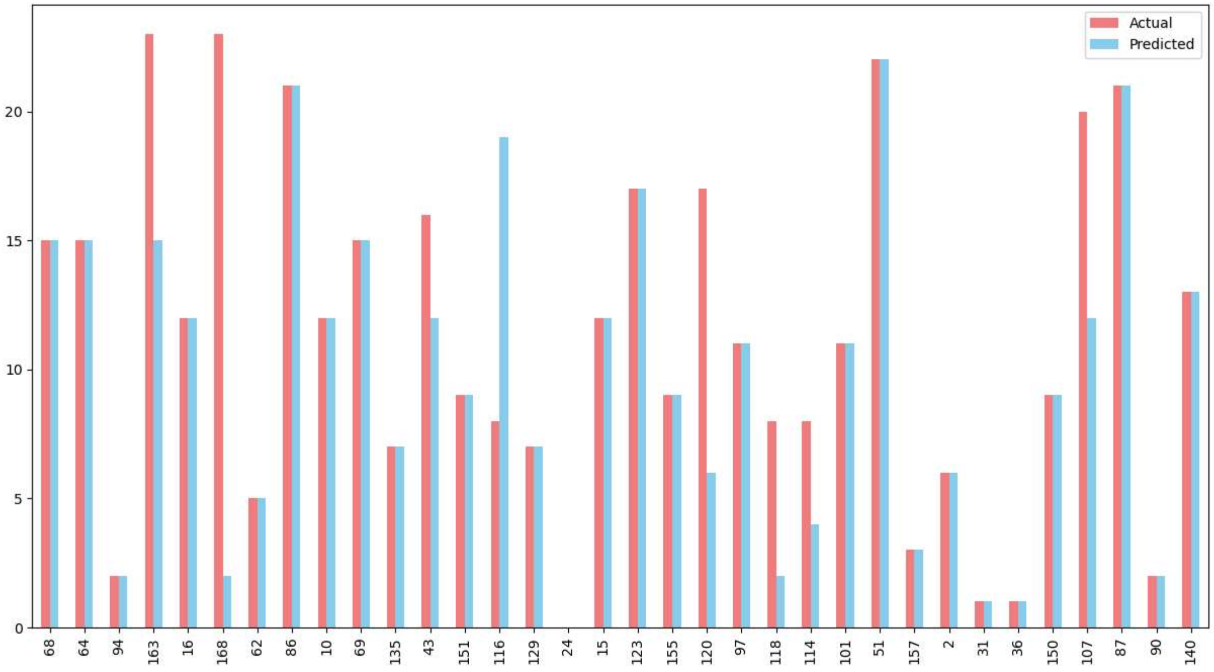
```
Accuracy Score:  0.7647058823529411
```

In [27]:

```
df_log = pd.DataFrame({'Actual': y_val, 'Predicted': y_pred_log_acc})
```

In [28]:

```
df_log.plot(kind = 'bar', figsize = (15, 8), color = ['lightcoral', 'skyblue'])
plt.show()
```



SVC

In [29]:

```
classifier = SVC()
classifier.fit(X_train, y_train)
```

Out[29]:

```
▾ SVC
SVC()
```

In [30]:

```
y_pred_svc = classifier.predict(X_val)
```

In [31]:

```
print('Accuracy of SVC Classifier: {:.2f}'.format(accuracy_score(y_val, y_pred_svc)
```

```
Accuracy of SVC Classifier: 0.29
```

Grid Search

In [32]:

```
param_grid = {'C':[0.1,1, 10, 100, 1000], 'gamma':[1, 0.1, 0.01, 0.001, 0.0001]}
```

In [33]:

```
grid_svc_acc = GridSearchCV(classifier, param_grid)
grid_svc_acc.fit(X_train, y_train)

#Predict values based on new parameters
y_pred_svc_acc = grid_svc_acc.predict(X_val)

#New Model EValuation metrics
print('Accuracy Score : ', accuracy_score(y_val, y_pred_svc_acc))
```

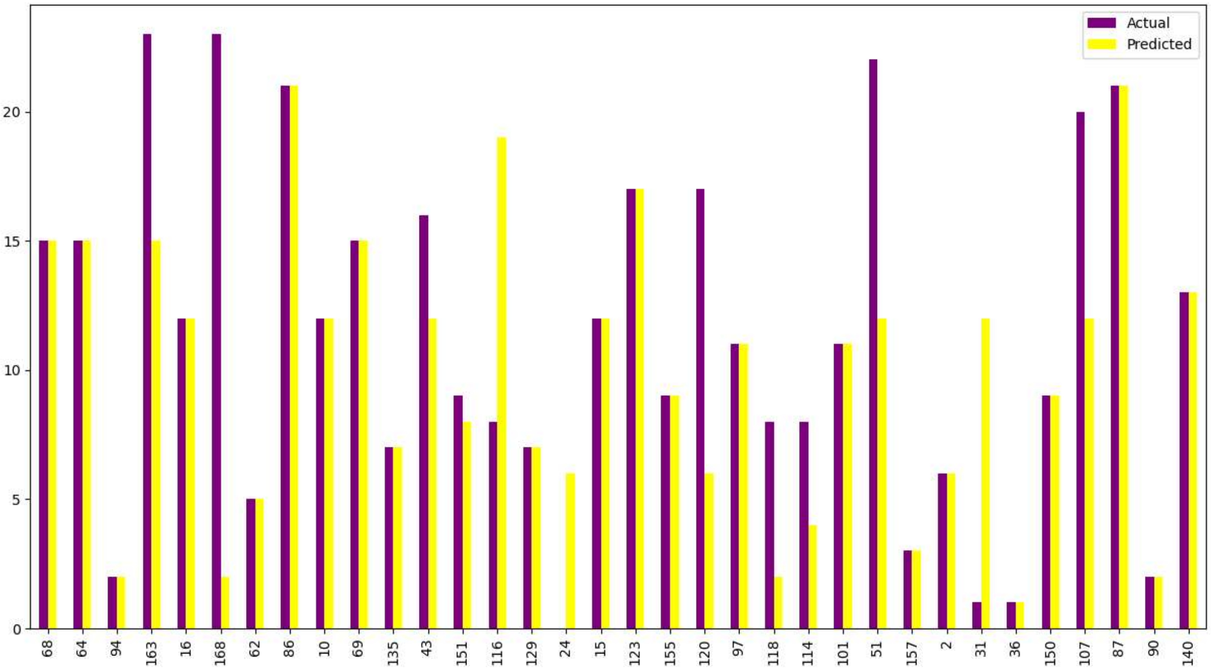
```
Accuracy Score :  0.6470588235294118
```

In [34]:

```
df_svc = pd.DataFrame({'Actual': y_val, 'Predicted': y_pred_svc_acc})
```

In [35]:

```
df_svc.plot(kind = 'bar', figsize = (15,8), color = ['purple', 'yellow'])
plt.show()
```



Random Forest

In [36]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [37]:

```
clf_rf = RandomForestClassifier(n_estimators = 200)
clf_rf.fit(X_train, y_train)
```

Out[37]:

```
▾ RandomForestClassifier
RandomForestClassifier(n_estimators=200)
```

In [38]:

```
y_pred_rf = clf_rf.predict(X_val)
```

In [39]:

```
print('Accuracy of SVC Classifier : {:.2f}'.format(accuracy_score(y_val, y_pred_rf)
```

```
Accuracy of SVC Classifier : 0.74
```

Grid Search

In [40]:

```
param_grid_rf = {"n_estimators": np.arange(2, 300, 2)}
```

In [41]:

```
from sklearn.model_selection import GridSearchCV
```

In [48]:

```
grid_rf_acc = GridSearchCV(clf_rf, param_grid_rf)
```

In [49]:

```
grid_rf_acc.fit(X_train, y_train)
```

Out[49]:

```
▸ GridSearchCV
▸ best_estimator_: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [50]:

```
#predict values based on new parameters
y_pred_rf_acc = grid_rf_acc.predict(X_val)

#New Model Evaluation Metrics
print('Accuracy Score : ', accuracy_score(y_val, y_pred_rf_acc))
```

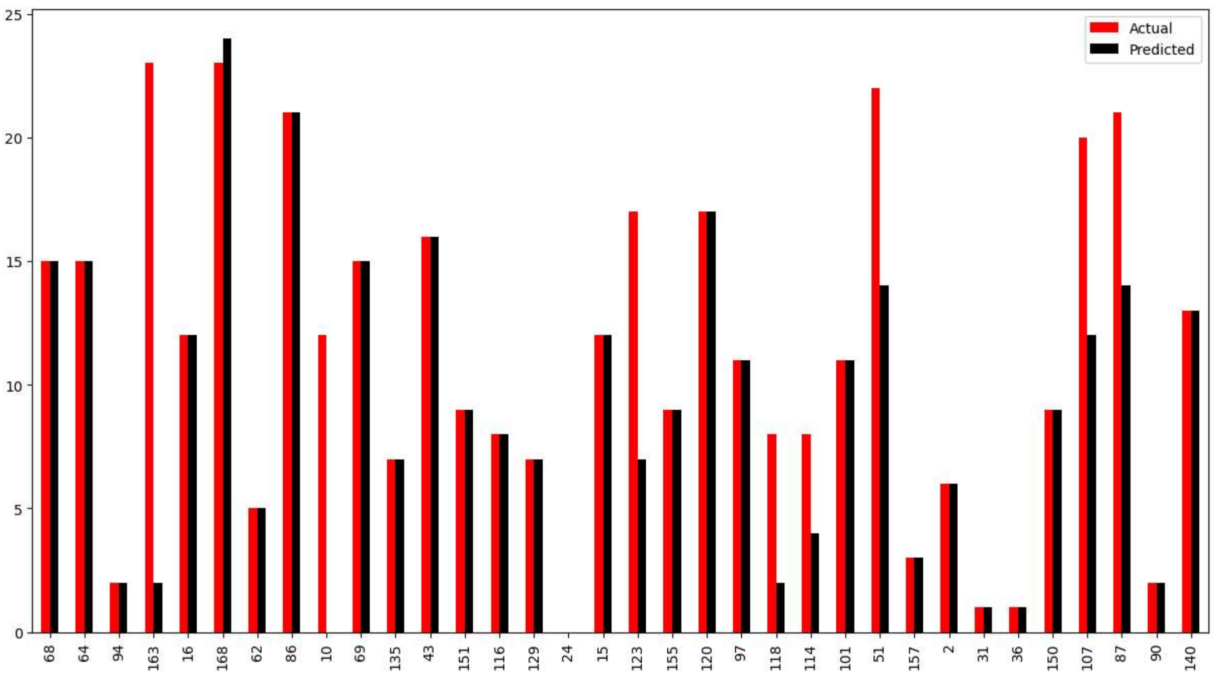
```
Accuracy Score :  0.7352941176470589
```

In [51]:

```
df_rf = pd.DataFrame({'Actual': y_val, 'Predicted': y_pred_rf_acc})
```

In [58]:

```
df_rf.plot(kind = 'bar', figsize = (15, 8), color = ['red', 'black'])
plt.show()
```



In [53]:
```
y_val_inverse = le.inverse_transform(y_val)
```

In [54]:
```
predicted_inverse = le.inverse_transform(y_pred_log_acc)
```

In [55]:
```
df_1 = pd.DataFrame({'Actual': y_val_inverse, 'Predicted': predicted_inverse})
df_1[1:7]
```

Out[55]:

	Actual	Predicted
1	Java Developer	Java Developer
2	Automation Testing	Automation Testing
3	Testing	Java Developer
4	HR	HR
5	Testing	Automation Testing
6	Civil Engineer	Civil Engineer

Single Prediction

In [60]:
```
text = "KEY SKILLS: Ã¢Â€Â¢ Computerized accounting with tally Ã¢Â€Â¢ Sincere & hard
```

In [61]:
```
text = get_clean(text)
print(text)
```

```
key skill computer account talli sincer hard work manag account incom tax good commun leadership two four wheeler drive licens internet ecommerc manag comput skill c languag web program talli dbm educ detail june june mba financ hr india mlrit june june bcom comput hyderabad telangana osmania univers june april inter mec india srimedhavhrnaniskil detail account exprienc month databas manag system exprienc month dbm exprienc month manag account exprienc month ecommerc exprienc monthscompani detail compani valuelab descript give rrf form requir dlt hand rlt scrum master take form rlt scrum master give form traine work requir till candid receiv offer compani
```

In [62]:
```python
vec = tfidf.transform([text])
```

In [63]:
```python
predict = grid_log_acc.predict(vec)
predict
```

Out[63]:
```
array([12])
```

In [66]:
```python
y_pred = le.inverse_transform(predict)
y_pred
```

Out[66]:
```
array(['HR'], dtype=object)
```

Another Example

In [67]:
```python
text_ds = "Machine learning, Deep learning, Python, Statistics"
```

In [68]:
```python
text_ds = get_clean(text_ds)
print(text_ds)
```

```
machin learn deep learn python statist
```

In [69]:
```python
vec = tfidf.transform([text_ds])
```

In [70]:
```python
predict = grid_log_acc.predict(vec)
predict
```

Out[70]:
```
array([6])
```

In [71]:
```python
y_pred = le.inverse_transform(predict)
y_pred
```

Out[71]:
```
array(['Data Science'], dtype=object)
```

In [ ]:
```python

```