Working on the training data

In [184…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import ConfusionMatrixDisplay

df=pd.read_csv('instagram_user.csv')
df=df.iloc[:,:12]
df.sample(5)
```

Out[184…

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	externa URl
151	1	0.15	2	0.0	0	37	(
344	0	0.00	2	0.0	0	0	(
201	1	0.00	0	0.0	0	8	(
236	1	0.00	2	0.0	0	0	(
310	0	0.57	1	0.0	0	0	(

In [163…

```
# check for the missing values
df.shape
```

Out[163…

```
(576, 12)
```

In [164…

```
df.describe()
```

Out[164…

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length
count	576.000000	576.000000	576.000000	576.000000	576.000000	576.000000
mean	0.701389	0.163837	1.460069	0.036094	0.034722	22.623264
std	0.458047	0.214096	1.052601	0.125121	0.183234	37.702987
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
75%	1.000000	0.310000	2.000000	0.000000	0.000000	34.000000
max	1.000000	0.920000	12.000000	1.000000	1.000000	150.000000

In [165…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 576 entries, 0 to 575
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   profile pic           576 non-null    int64  
 1   nums/length username  576 non-null    float64
 2   fullname words        576 non-null    int64  
 3   nums/length fullname  576 non-null    float64
 4   name==username        576 non-null    int64  
 5   description length    576 non-null    int64  
 6   external URL          576 non-null    int64  
 7   private               576 non-null    int64  
 8   #posts                576 non-null    int64  
 9   #followers            576 non-null    int64  
 10  #follows              576 non-null    int64  
 11  fake                  576 non-null    int64  
dtypes: float64(2), int64(10)
memory usage: 54.1 KB
```

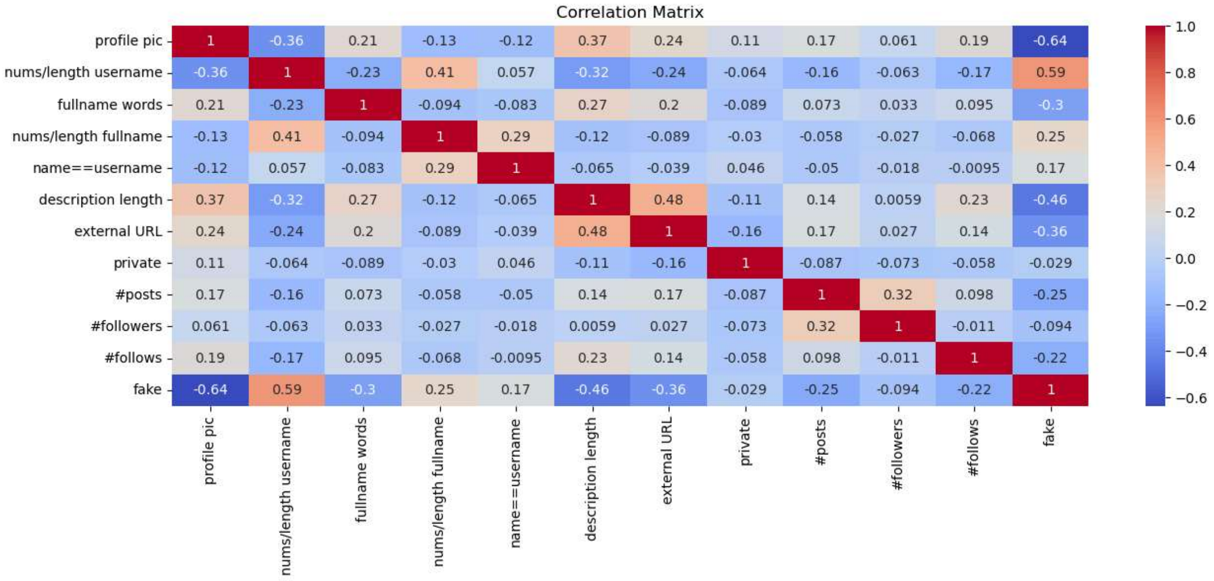
In [166…

```
# number of unique values in dataframe
df.nunique()
```

```
Out[166…  profile pic               2
          nums/length username     54
          fullname words            9
          nums/length fullname     25
          name==username            2
          description length      104
          external URL              2
          private                   2
          #posts                  193
          #followers              372
          #follows                400
          fake                      2
          dtype: int64
```

- CORRELATION MATRIX BETWEEN EACH FEATURE

```
In [168…  correlation=df.corr()
          plt.subplots(figsize=(15,5))
          sns.heatmap(correlation,cmap='coolwarm',annot=True)
          plt.title('Correlation Matrix')
          plt.show()
          correlation
```



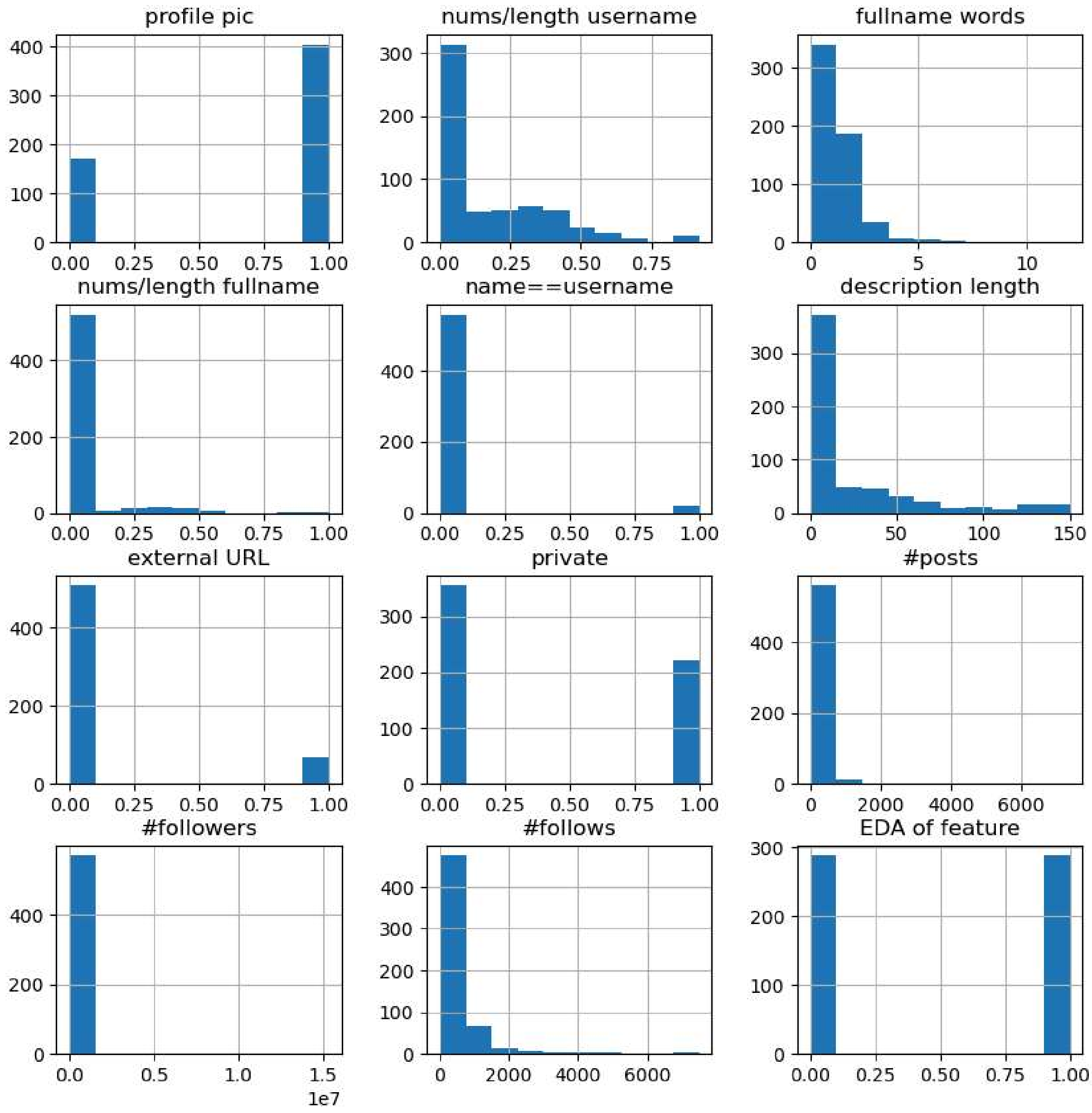
Out[168…

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	des
profile pic	1.000000	-0.364087	0.213295	-0.131756	-0.124903	(
nums/length username	-0.364087	1.000000	-0.225472	0.408567	0.056890	-(
fullname words	0.213295	-0.225472	1.000000	-0.094348	-0.082969	(
nums/length fullname	-0.131756	0.408567	-0.094348	1.000000	0.291149	-(
name==username	-0.124903	0.056890	-0.082969	0.291149	1.000000	-(
description length	0.367892	-0.321170	0.272522	-0.117521	-0.064814	'
external URL	0.236729	-0.237125	0.196562	-0.088724	-0.039232	(
private	0.114732	-0.063713	-0.089070	-0.030030	0.046084	-(
#posts	0.169570	-0.157442	0.073350	-0.057716	-0.049808	(
#followers	0.061137	-0.062785	0.033225	-0.027035	-0.017761	(
#follows	0.194833	-0.172413	0.094855	-0.067971	-0.009529	(
fake	-0.637315	0.587687	-0.298793	0.246782	0.170695	-(

EDA

In [170…

```
df.hist(figsize=(10,10))
plt.title('EDA of feature')
plt.show()
```

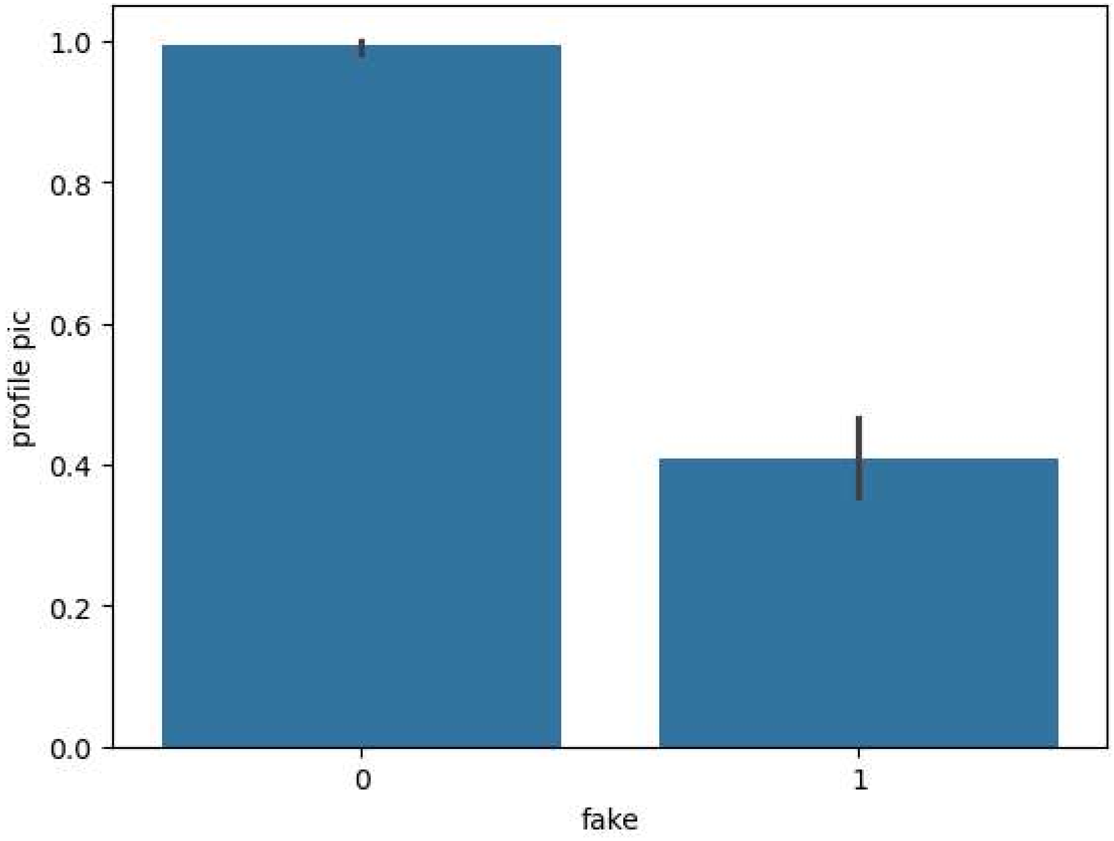


- Profile picture (Fake vs genuine)

In [172…

```
print(df['profile pic'].value_counts())
sns.barplot(x='fake', y='profile pic', data=df)
plt.show()
```

```
profile pic
1    404
0    172
Name: count, dtype: int64
```

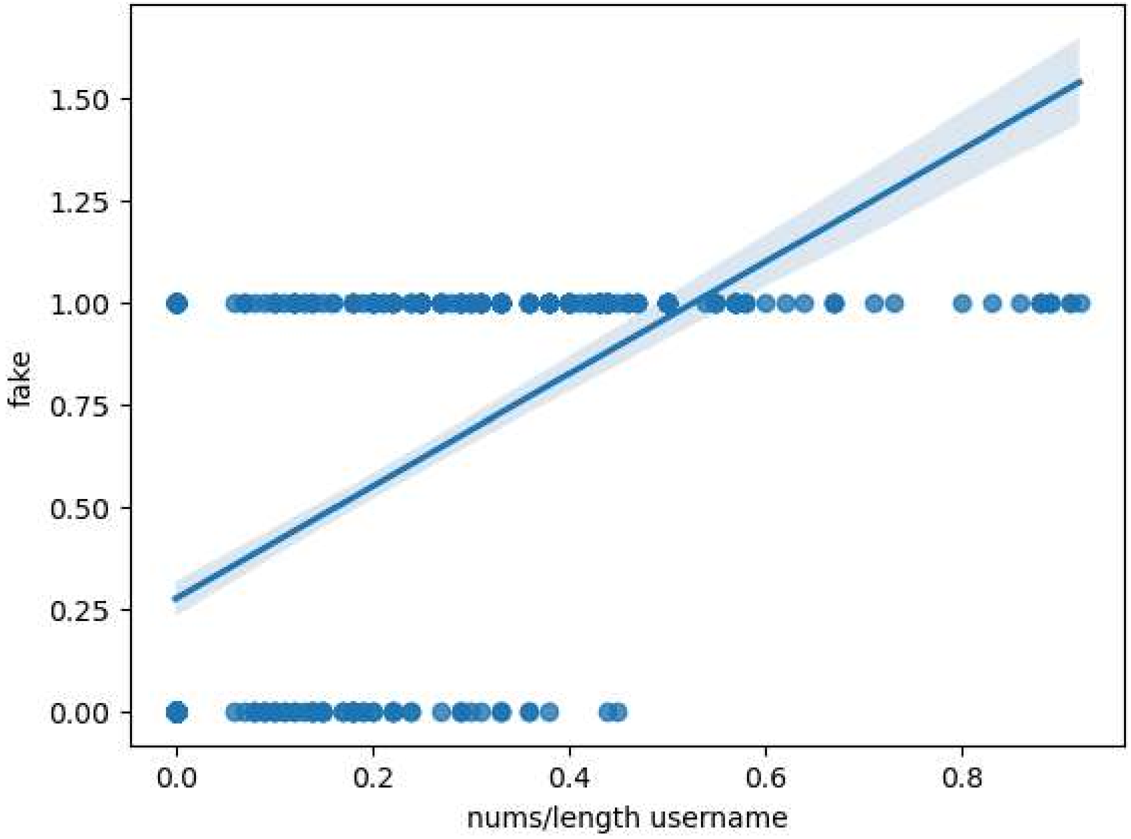


In [173…

```
sns.regplot(data=df,x='nums/length username',y='fake')
```

Out[173…

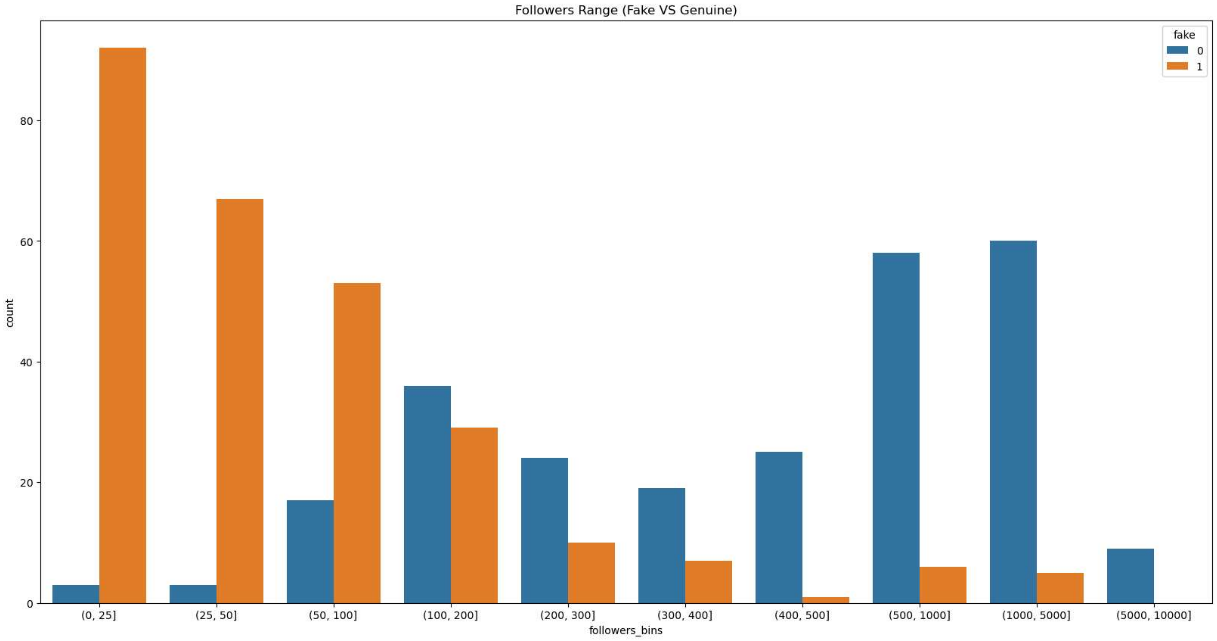
```
<Axes: xlabel='nums/length username', ylabel='fake'>
```



- Followers VS fake

In [175…

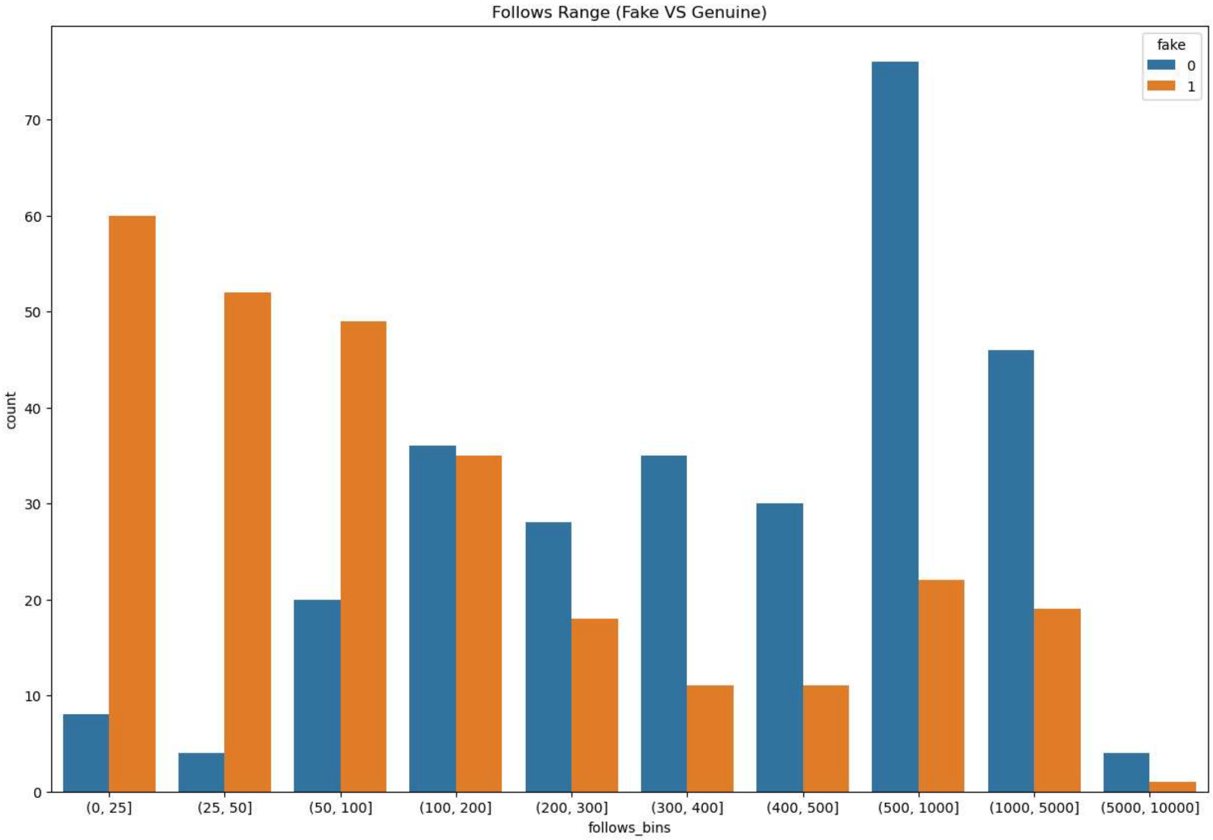
```
# Followers column
bins=[0,25,50,100,200,300,400,500,1000,5000,10000]
df['followers_bins']=pd.cut(df['#followers'],bins=bins)
plt.subplots(figsize=(20,10))
plt.title('Followers Range (Fake VS Genuine)')
sns.countplot(data=df,x='followers_bins',hue='fake')
plt.show()
```



- Follows which are fake

In [177…

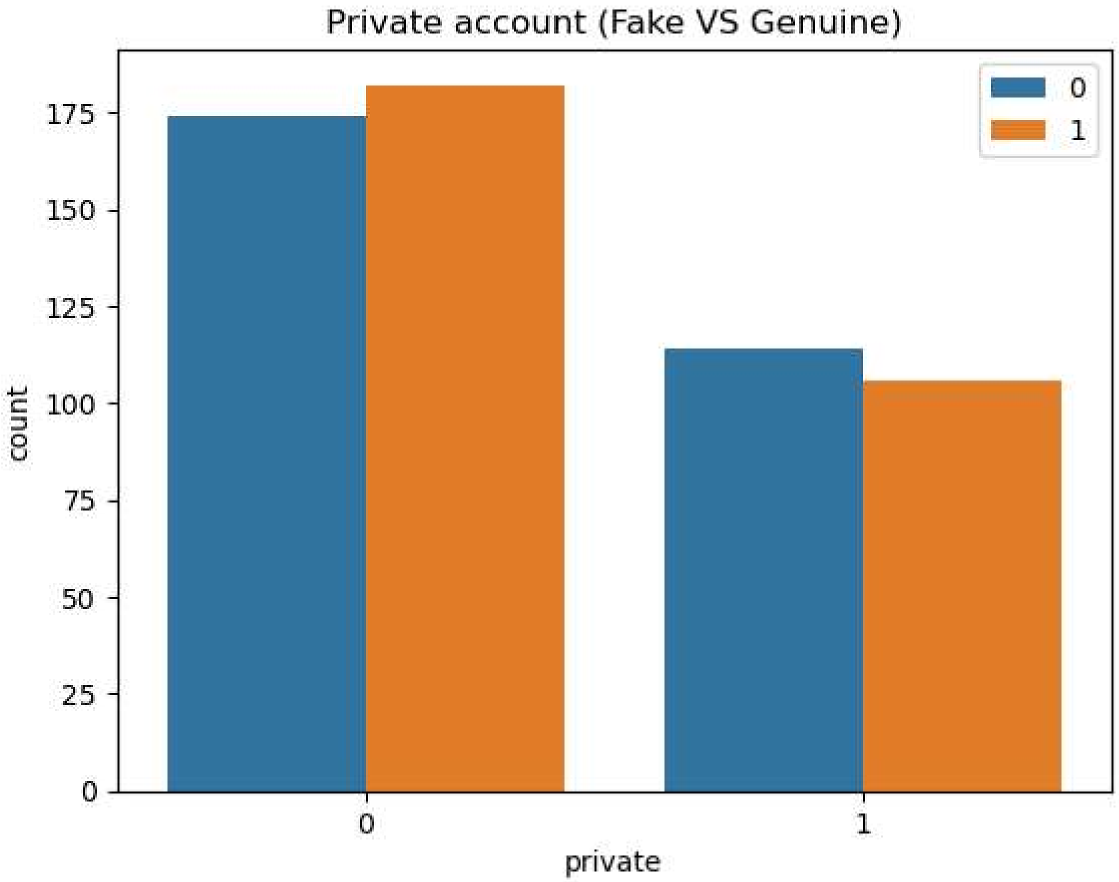
```
# Follows column which are fake
bins=[0,25,50,100,200,300,400,500,1000,5000,10000]
df['follows_bins']=pd.cut(df['#follows'],bins=bins)
plt.subplots(figsize=(15,10))
plt.title('Follows Range (Fake VS Genuine)')
sns.countplot(data=df,x='follows_bins',hue='fake')
plt.show()
```



- Private account (Fake vs Genuine)

In [179…

```
sns.countplot(data=df,x='private',hue='fake')
plt.title('Private account (Fake VS Genuine)')
plt.legend()
plt.show()
```



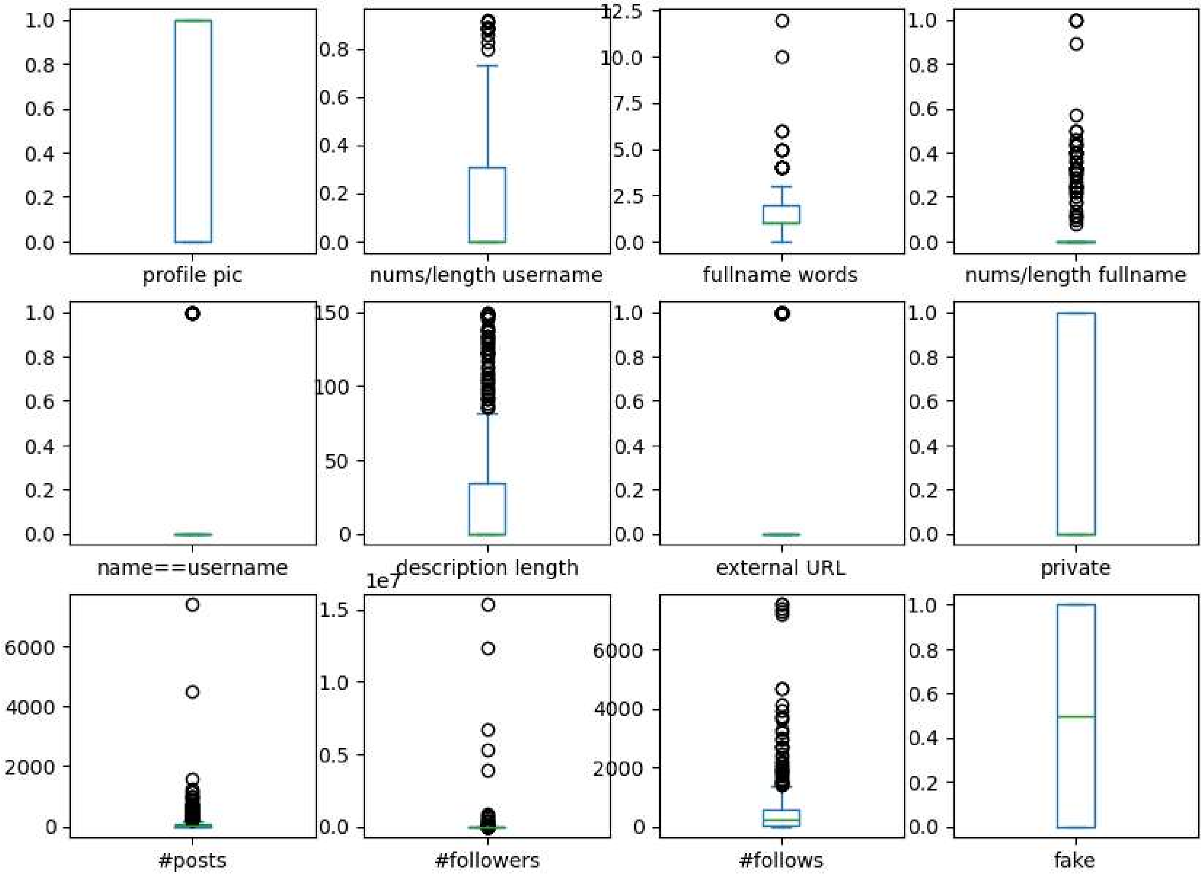
- outlier detection

In [181…

```
df.plot(kind='box',subplots=True,layout=(4,4),figsize=(10,10))
```

Out[181…

```
profile pic               Axes(0.125,0.712609;0.168478x0.167391)
nums/length username    Axes(0.327174,0.712609;0.168478x0.167391)
fullname words          Axes(0.529348,0.712609;0.168478x0.167391)
nums/length fullname    Axes(0.731522,0.712609;0.168478x0.167391)
name==username            Axes(0.125,0.511739;0.168478x0.167391)
description length      Axes(0.327174,0.511739;0.168478x0.167391)
external URL            Axes(0.529348,0.511739;0.168478x0.167391)
private                 Axes(0.731522,0.511739;0.168478x0.167391)
#posts                     Axes(0.125,0.31087;0.168478x0.167391)
#followers               Axes(0.327174,0.31087;0.168478x0.167391)
#follows                 Axes(0.529348,0.31087;0.168478x0.167391)
fake                     Axes(0.731522,0.31087;0.168478x0.167391)
dtype: object
```



calculating accuracy score

In [186…

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
x=df.drop(columns=['fake'])
y=df['fake']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)

#Standardization
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)

# Decisiontree Classifier
clf=RandomForestClassifier(n_estimators=100,random_state=42)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)

#GridsearchCV
param_grid={
    'n_estimators':[50,100,200],
    'max_depth':[None,10,20],
    'min_samples_split':[2,5]
}
grid_search=GridSearchCV(estimator=clf,param_grid=param_grid,cv=10,scoring='accurac
#calculatin accuracy score
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_pred,y_test)*100
print('accuracy before cross_val_cscore: ',accuracy)

# cross_val_score
from sklearn.model_selection import cross_val_score
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
print('Accuracy after cross_val_score',np.mean(cross_val_score(clf,x,y,cv=25,scorin
```

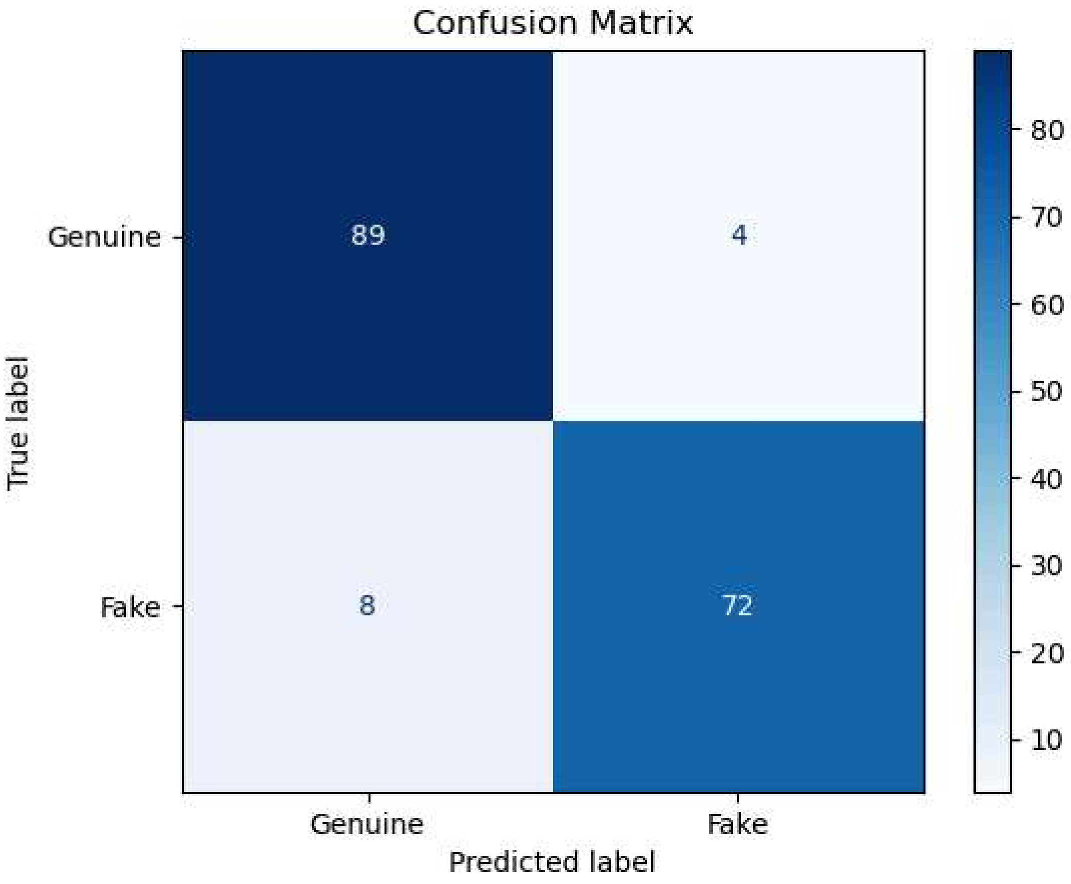
```
accuracy before cross_val_cscore:  93.0635838150289
Accuracy after cross_val_score 88.52898550724639
```

visualize confusion matrix

In [188…

```
from sklearn.metrics import confusion_matrix,classification_report
ConfusionMatrixDisplay.from_predictions(y_test, y_pred,
display_labels=['Genuine', 'Fake'], cmap='Blues')
plt.title("Confusion Matrix")
plt.show()

print('Confusion Matrix:\n',confusion_matrix(y_pred,y_test))
print('Classification Report:\n',classification_report(y_pred,y_test))
```



```
Confusion Matrix:
 [[89  8]
 [ 4 72]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.92      0.94        97
           1       0.90      0.95      0.92        76

    accuracy                           0.93       173
   macro avg       0.93      0.93      0.93       173
weighted avg       0.93      0.93      0.93       173
```

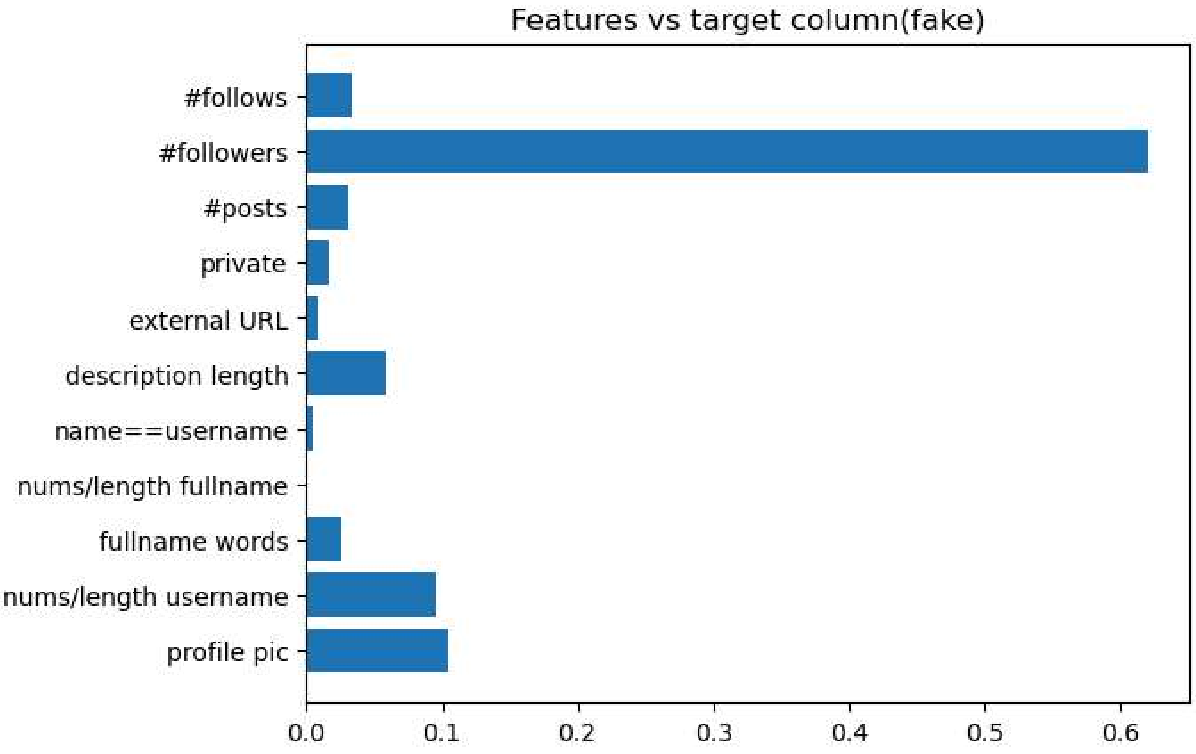
plotting feature importance

In [191…

```
plt.barh(x.columns,clf.feature_importances_)
plt.title('Features vs target column(fake)')
# It shows that '#followers' column is most proportional to the detection of fake
```

Out[191…

```
Text(0.5, 1.0, 'Features vs target column(fake)')
```



In [193…

```
# Final dataset would look as :
df.sample(5)
```

Out[193…

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	externa URl
102	1	0.00	3	0.0	0	0	0
555	1	0.46	1	0.5	0	0	0
130	1	0.00	4	0.0	0	150	1
485	0	0.44	1	0.0	0	0	0
308	0	0.45	3	0.0	0	0	0

In [ ]: