PCA and SVD

```
X = df.drop('fake', axis=1)
y = df['fake']

# Perform PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Perform SVD
svd = TruncatedSVD(n_components=2)
X_svd = svd.fit_transform(X)

# Print the explained variance ratios
print("Explained variance ratio (PCA):", pca.explained_variance_ratio_)
print("Explained variance ratio (SVD):", svd.explained_variance_ratio_)
```

```
Explained variance ratio (PCA): [9.99998806e-01 1.01968536e-06]
Explained variance ratio (SVD): [9.99998803e-01 1.02129951e-06]
```

```
X_train, X_test, y_train, y_test = train_test_split(df.drop('fake', axis=1), df['fake'], test_size=0.2, random_state=42)

print('X_train : ')
print(X_train.head())
print('')
print('X_test : ')
print(X_test.head())
print('')
print('y_train : ')
print(y_train.head())
print('')
print('y_test : ')
print(y_test.head())
```

```
     profile pic  nums/length username  fullname words  nums/length fullname  \
437            0                   0.4               1                   0.0
63             1                   0.0               1                   0.0
208            1                   0.0               0                   0.0
60             1                   0.0               1                   0.0
15             1                   0.0               3                   0.0

     name==username  description length  external URL  private  #posts  \
437               0                   0             0        1       0
63                0                  68             1        0     100
208               0                  18             0        1     133
60                0                   3             0        0      12
15                0                  46             0        0     254

     #followers  #follows
437          24        88
63          802       151
208        1008       517
60          108        97
15        50374       900

X_test :
     profile pic  nums/length username  fullname words  nums/length fullname  \
234            1                  0.17               2                  0.00
118            1                  0.00               0                  0.00
346            1                  0.31               1                  0.31
498            0                  0.28               1                  0.24
402            1                  0.00               2                  0.00

     name==username  description length  external URL  private  #posts  \
234               0                 134             1        0      31
118               0                 123             1        0     107
346               1                   0             0        1       0
498               0                   0             0        0       0
402               0                   0             0        0      14

     #followers  #follows
234         265       321
118         971      2047
346          25        86
498          86         0
402         143       500

y_train :
437    1
63     0
208    0
60     0
```

```
Name: Fake, dtype: int64

y_test :
234    0
118    0
346    1
498    1
402    1
Name: fake, dtype: int64
```

```
#scaling the values into the range 0-1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df= pd.DataFrame(scaler.fit_transform(df))
df
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	0.293478	0.000000	0.00	0.0	0.353333	0.0	0.0	0.004331	0.000065	0.127333	0.0
1	1.0	0.000000	0.166667	0.00	0.0	0.293333	0.0	0.0	0.038706	0.000179	0.071067	0.0
2	1.0	0.108696	0.166667	0.00	0.0	0.000000	0.0	1.0	0.001759	0.000010	0.013067	0.0
3	1.0	0.000000	0.083333	0.00	0.0	0.546667	0.0	0.0	0.091893	0.000027	0.086800	0.0
4	1.0	0.000000	0.166667	0.00	0.0	0.000000	0.0	1.0	0.000812	0.000010	0.016800	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
571	1.0	0.597826	0.083333	0.44	0.0	0.000000	0.0	0.0	0.004466	0.000011	0.079467	1.0
572	1.0	0.413043	0.083333	0.33	0.0	0.140000	0.0	0.0	0.005955	0.000004	0.010000	1.0
573	1.0	0.619565	0.166667	0.00	0.0	0.000000	0.0	0.0	0.000541	0.000006	0.045200	1.0
574	1.0	0.619565	0.083333	0.00	0.0	0.073333	0.0	0.0	0.000000	0.000004	0.009733	1.0
575	1.0	0.293478	0.083333	0.00	0.0	0.000000	0.0	0.0	0.000271	0.000010	0.064933	1.0

576 rows × 12 columns

Decision Tree

```
#building the Decision Tree classifier
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
model = dt.fit(X_train, y_train)
```

```
from sklearn.tree import DecisionTreeRegressor
```

```
reg = DecisionTreeRegressor(max_depth=2)
```

```
reg.fit(X, y)
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=2,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, #min_impurity_split=None,

min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, #presort='deprecated',
random_state=None, splitter='best')
```

```
DecisionTreeRegressor
DecisionTreeRegressor(criterion='mse', max_depth=2)
```

```
from sklearn.tree import plot_tree
```

```
plot_tree(reg)
```

```
[Text(0.5, 0.75, 'x[7] <= 0.5\nsquared_error = 0.213\nsamples = 120\nvalue = 0.308'),
 Text(0.25, 0.25, 'squared_error = 0.0\nsamples = 83\nvalue = 0.0'),
 Text(0.75, 0.25, 'squared_error = 0.0\nsamples = 37\nvalue = 1.0')]
```

```
y_hat = dt.predict(X_test)
```

```
|squared_error = 0.213|
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_hat)
```

```
0.8706896551724138
```

```
|squared_error = 0.0|  |squared_error = 0.0|
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression
LogisticRegression()

```
# predicting the test set results
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print('R-squared value:', r2)
```

```
R-squared value: 0.6178496555855046
```

Linear Regression

```
# create linear regression model
from sklearn import linear_model, metrics
model = linear_model.LinearRegression()
```

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

▾ LinearRegression
LinearRegression()

```
print('Coefficients: \n', model.coef_)
```

```
Coefficients:
 [-4.48562952e-01  7.97309015e-01 -3.62284700e-02 -9.56923547e-02
  2.36095048e-01 -1.69006009e-03 -1.31382930e-01  9.56927289e-03
 -8.01725424e-05 -1.06252148e-08 -2.00946860e-05]
```

```
# predicting the test set results
y_pred = model.predict(X_test)
```

```
# Results of Linear Regression.
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print("Mean Square Error : ", mse)
```

```
Mean Square Error :  0.11467750277173532
```

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print('R-squared value:', r2)
```

```
R-squared value: 0.5378555024568823
```

Logistic Regression