```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_recall_fscore_support
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import FunctionTransformer
```

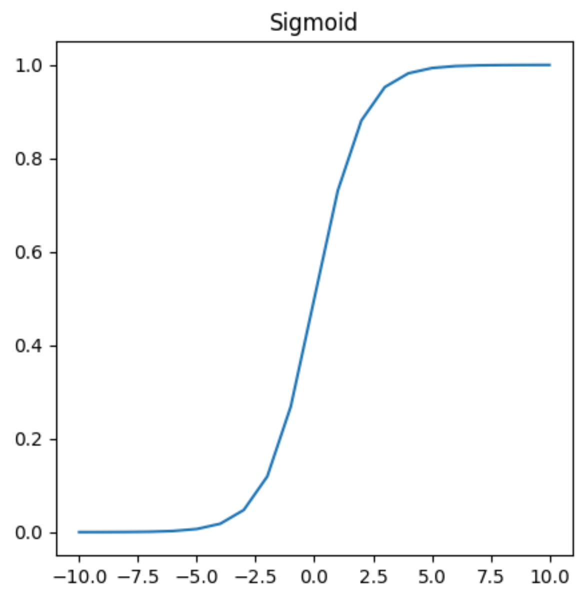
```
a =np.array([x for x in range(-10,11)])
```

```
sigmoid =FunctionTransformer(lambda x: 1 / (1 + np.exp(-x)))
relu = FunctionTransformer(lambda x: np.maximum(0, x))
tanh = FunctionTransformer(lambda x: np.tanh(x))
```

```
X = sigmoid.transform(a)
Y = relu.transform(a)
Z = tanh.transform(a)
```

```
plt.figure(figsize=(5,5))
plt.plot(a,X)
plt.title('Sigmoid')
```

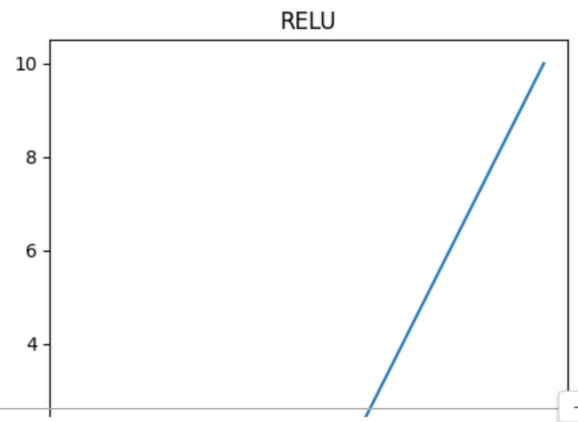
```
Text(0.5, 1.0, 'Sigmoid')
```



Normalizes between 0 and 1

```
plt.figure(figsize=(5,5))
plt.plot(a,Y)
plt.title('RELU')
```

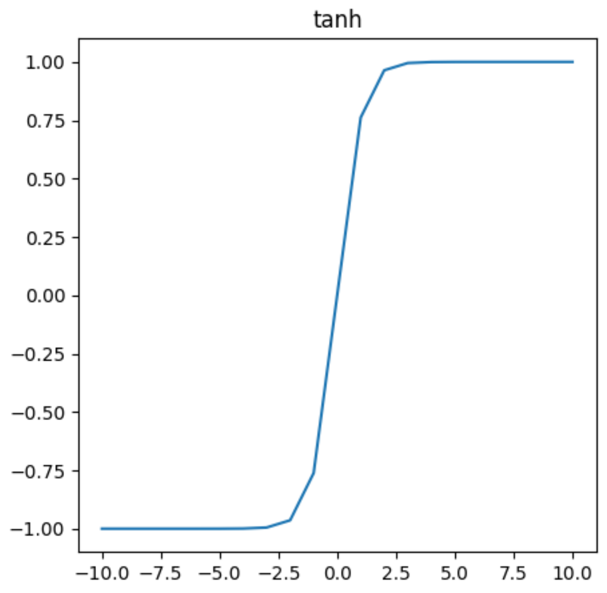
```
Text(0.5, 1.0, 'RELU')
```



cut off negative values

```
plt.figure(figsize=(5,5))
plt.plot(a,Z)
plt.title('tanh')
```

```
Text(0.5, 1.0, 'tanh')
```



normalizes between -1 to 1

Performance Metrics

Calculate Metrics For **KNN Task**

```
data=load_wine()
X = pd.DataFrame(data=data['data'], columns=data['feature_names'])
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train,y_train);
y_pred = classifier.predict(X_test)
score =classifier.score(X_test,y_test)
print(score)
```

```
0.8055555555555556
```

```
pd.DataFrame(metrics.classification_report(y_test, y_pred, output_dict=True)).transpose()
```

	precision	recall	f1-score	support
0	1.0	1.0	1.0	14.0
1	1.0	1.0	1.0	14.0
2	1.0	1.0	1.0	8.0
accuracy	1.0	1.0	1.0	1.0
macro avg	1.0	1.0	1.0	36.0
weighted avg	1.0	1.0	1.0	36.0

Double-click (or enter) to edit

Calculate metrics for Random Forest Task

```
classifier=RandomForestClassifier(criterion='gini',bootstrap=True)
classifier.fit(X_train,y_train);
y_pred=classifier.predict(X_test)
print(classifier.score(X_test,y_test))
pd.DataFrame(metrics.classification_report(y_test, y_pred, output_dict=True)).transpose()
```

```
1.0
```

	precision	recall	f1-score	support
0	1.0	1.0	1.0	14.0
1	1.0	1.0	1.0	14.0
2	1.0	1.0	1.0	8.0
accuracy	1.0	1.0	1.0	1.0
macro avg	1.0	1.0	1.0	36.0
weighted avg	1.0	1.0	1.0	36.0