Home | Assignment Classification problem

localhost:8888/notebooks/Assignment%20Classification%20problem.ipynb?

jupyter Assignment Classification problem Last Checkpoint: 2 days ago

Trusted

File Edit View Run Kernel Settings Help

Markdown | JupyterLab | Python 3 (ipykernel)

CLASSIFICATION

[ ]:
```
#Objective of this assignment is to evaluate your understanding and ability to apply supervised learning techniques to areal world Dataset
```

[ ]:
```
#Use the Breast cancer dataset available in the sklearn library
```

Loading and Preprocessing

[104]:
```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[106]:
```
#Load the breast cancer dataset
Data = load_breast_cancer()
```

[108]:
```
#Convert the dataset to a pandas dataframe
df= pd.DataFrame( Data.data ,columns = Data.feature_names)
df['target'] = Data.target
```

[110]:
```
df.head()
```

[110]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245

Nifty midcap -1.27%

Search

ENG IN 10:56 22-01-2025

Home | Assignment Classification problem

localhost:8888/notebooks/Assignment%20Classification%20problem.ipynb?

jupyter Assignment Classification problem Last Checkpoint: 2 days ago

Trusted

File Edit View Run Kernel Settings Help

Markdown | JupyterLab | Python 3 (ipykernel)

[110]:
```
df.head()
```

[110]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050

5 rows × 31 columns

[114]:
```
df.tail()
```

[114]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2027.0	0.14100	0.2113
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1731.0	0.11660	0.1922
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1124.0	0.11390	0.3094
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1821.0	0.16500	0.8681
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	268.6	0.08996	0.0644

5 rows × 31 columns

Nifty midcap -1.27%

Search

ENG IN

10:57 22-01-2025

Home | Assignment Classification problem

localhost:8888/notebooks/Assignment%20Classification%20problem.ipynb?

jupyter Assignment Classification problem Last Checkpoint: 2 days ago

Trusted

File Edit View Run Kernel Settings Help

Markdown | JupyterLab | Python 3 (ipykernel)

[112]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
 21  worst texture           569 non-null    float64
 22  worst perimeter         569 non-null    float64
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
```

Nifty midcap -1.27%

Search

ENG IN 10:57 22-01-2025

Home | Assignment Classification problem

localhost:8888/notebooks/Assignment%20Classification%20problem.ipynb?

jupyter Assignment Classification problem Last Checkpoint: 2 days ago

Trusted

File Edit View Run Kernel Settings Help

Markdown | JupyterLab | Python 3 (ipykernel)

[52]:
```
#Split the data into features(X) and target(Y)
#X = df.drop('target' , axis =1)
#Y = df['target']
```

[118]:
```
#Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['target'] = le.fit_transform(df['target'])
```

[62]:
```
#Handling missing values
print("Missing Values count:" , df.isnull().sum())
```

```
Missing Values count: mean radius               0
mean texture              0
mean perimeter            0
mean area                 0
mean smoothness           0
mean compactness          0
mean concavity            0
mean concave points       0
mean symmetry             0
mean fractal dimension    0
radius error              0
texture error             0
perimeter error           0
area error                0
smoothness error          0
compactness error         0
concavity error           0
concave points error      0
symmetry error            0
fractal dimension error   0
worst radius              0
worst texture             0
worst perimeter           0
worst area                0
```

Nifty midcap -1.27%

Search

ENG IN 10:57 22-01-2025

```
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

[10]:
```
#Perform feature scaling using standard scaler
Scaler = StandardScaler()
Scaled_data = Scaler.fit_transform(df[Data.feature_names])
Scaled_data
```

[10]:
```
array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])
```

[12]:
```
#Split the data into training and testing sets
X_train,X_test ,Y_train,Y_test = train_test_split(df[Data.feature_names], df['target'],test_size = 0.2 ,random_state=42)
```

[ ]:
```
#Explain the preprocessing step performed and why they are necessary for this dataset
```

Preprocessing steps Performed

1.Checking for missing values:The breast cancer dataset does not contain any missing values. So no imputation is necessary
2.Encoding : In this dataset the target variable is already numerical as 0 and 1. So,no encoding is needed.
3.Feature Scaling : This dataset contains features with different scales ,Which can affect the performance of some classification algorithms. StandardScaler is used to standardize the features to have zero mean and unit vaiance.

1.Checking for missing values:The breast cancer dataset does not contain any missing values. So no imputation is necessary
2.Encoding : In this dataset the target variable is already numerical as 0 and 1. So,no encoding is needed.
3.Feature Scaling : This dataset contains features with different scales ,Which can affect the performance of some classification algorithms. StandardScaler is used to standardize the features to have zero mean and unit vaiance.

Classification Algorithm Implementation

1.Logistic Regression

[16]:
```
import warnings
warnings.filterwarnings("ignore")
```

[18]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(X_train,Y_train)
```

[18]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

[20]:
```
y_predict_logreg = model.predict(X_test)
y_predict_logreg
```

[20]:
```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0])
```

Logistic Regression is a suitable algorithm for this dataset because it is a binary classification problem and logistic regression is a popular choice for binary classification tasks.

2.Decision Tree Classifier

[39]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train,Y_train)
```

[39]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

[22]:
```
y_predict_dt = model.predict(X_test)
y_predict_dt
```

[22]:
```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0])
```

```
Decision Tree Classifier is a suitable algorithm for this dataset because it can handle high dimensional data and is
relatively easy to interpret.
```

3.Random Forest Classifier

[24]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train , Y_train)
```

[24]:
```
RandomForestClassifier
RandomForestClassifier()
```

4.Support Vector Machine

[28]:
```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
```

[28]:
```
SVC()
```

[32]:
```
y_predict_svm = model.predict(X_test)
y_predict_svm
```

[32]:
```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0])
```

SVM is a suitable algorithm for this dataset because it can handle high dimensional data and is a popular choice for binary calssification tasks.

5.K - Nearest Neighbours (K-NN)

[98]:
```
from sklearn.neighbors import KNeighborsClassifier
model  =KNeighborsClassifier()
model.fit(X_train,Y_train)
```

[98]:
```
KNeighborsClassifier()
```

[100]:
```
y_predict_Knn = model.predict(X_test)
```

5.K - Nearest Neighbours (K-NN)

[98]:
```
from sklearn.neighbors import KNeighborsClassifier
model  =KNeighborsClassifier()
model.fit(X_train,Y_train)
```

[98]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

[100]:
```
y_predict_Knn = model.predict(X_test)
y_predict_Knn
```

[100]:
```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0])
```

K-NN is a suitable algorithm for this dataset because it is a simple and intuitive algorithm that provide good performance on binary classification tasks.

Model Comparison

[38]:
```
from sklearn.metrics import accuracy_score
```

[58]:
```
#Calculate Accuracy of each model
accuracy_logreg = accuracy_score(Y_test,y_predict_logreg)
print("Accuracy of Logistic Regression:",accuracy_logreg)
```

```
Accuracy of Logistic Regression: 0.956140350877193
```

[60]:
```
accuracy_dt = accuracy_score(Y_test,y_predict_dt)
```

Model Comparison

[38]:
```
from sklearn.metrics import accuracy_score
```

[58]:
```
#Calculate Accuracy of each model
accuracy_logreg = accuracy_score(Y_test,y_predict_logreg)
print("Accuracy of Logistic Regression:",accuracy_logreg)
```

```
Accuracy of Logistic Regression: 0.956140350877193
```

[60]:
```
accuracy_dt = accuracy_score(Y_test,y_predict_dt)
print("Accuracy of Decision Tree Classifier:",accuracy_dt)
```

```
Accuracy of Decision Tree Classifier: 0.956140350877193
```

[82]:
```
accuracy_rf = accuracy_score(Y_test,y_predict_rf)
print("Accuracy of randomforest Classifier:",accuracy_rf)
```

```
Accuracy of randomforest Classifier: 0.9649122807017544
```

[64]:
```
accuracy_svm = accuracy_score(Y_test,y_predict_svm)
print("Accuracy of Support Vector Machine:",accuracy_svm)
```

```
Accuracy of Support Vector Machine: 0.9473684210526315
```

[102]:
```
accuracy_Knn =accuracy_score(Y_test,y_predict_Knn)
print("Accuracy of K-Nearest Neighbors:",accuracy_Knn)
```

```
Accuracy of K-Nearest Neighbors: 0.956140350877193
```

[78]:
```
#Determine the best Performing Model
best_model=max(accuracy_logreg,accuracy_dt,accuracy_rf,accuracy_svm,accuracy_Knn)
print("Best Performing Model:",best_model)
```

```
Best Performing Model: 0.9649122807017544
```

```
Based on accuarcy score ,
```

jupyter Assignment Classification problem Last Checkpoint: 2 days ago

Trusted

File Edit View Run Kernel Settings Help

Raw JupyterLab Python 3 (ipykernel)

```
Accuracy of Logistic Regression: 0.956140350877193
```

[60]:
```
accuracy_dt = accuracy_score(Y_test,y_predict_dt)
print("Accuracy of Decision Tree Classifier:",accuracy_dt)
```

```
Accuracy of Decision Tree Classifier: 0.956140350877193
```

[82]:
```
accuracy_rf = accuracy_score(Y_test,y_predict_rf)
print("Accuracy of randomforest Classifier:",accuracy_rf)
```

```
Accuracy of randomforest Classifier: 0.9649122807017544
```

[64]:
```
accuracy_svm = accuracy_score(Y_test,y_predict_svm)
print("Accuracy of Support Vector Machine:",accuracy_svm)
```

```
Accuracy of Support Vector Machine: 0.9473684210526315
```

[102]:
```
accuracy_Knn =accuracy_score(Y_test,y_predict_Knn)
print("Accuracy of K-Nearest Neighbors:",accuracy_Knn)
```

```
Accuracy of K-Nearest Neighbors: 0.956140350877193
```

[78]:
```
#Determine the best Performing Model
best_model=max(accuracy_logreg,accuracy_dt,accuracy_rf,accuracy_svm,accuracy_Knn)
print("Best Performing Model:",best_model)
```

```
Best Performing Model: 0.9649122807017544
```

```
Based on accuarcy score ,
The best performing model is Random Forest Classifier with an highest accuarcy of 0.96 .
The worst performing model is Support Vector Machine with a smallest accuarcy of 0.94.
(When KNN=4 , K-nearest Neighbor classifier performs as the worst model with accuracy 0.93)
```

```
The difference in accuracy between the models is relatively small .This suggests that all models are
performing reasonably well on this dataset.
```

[ ]: