Files

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

```
df=pd.read_csv("/content/drive/MyDrive/Breast_Cancer.csv")
```

```
df.head()
```

	id	diagnosis	radius_mean	texture_mean
0	842302	M	17.99	10.38
1	842517	M	20.57	17.77
2	84300903	M	19.69	21.25
3	84348301	M	11.42	20.38
4	84358402	M	20.29	14.34

5 rows × 32 columns

```
df.tail()
```

	id	diagnosis	radius_mean	texture_mean
564	926424	M	21.56	22.39
565	926682	M	20.13	28.25
566	926954	M	16.60	28.08
567	927241	M	20.60	29.33
568	92751	B	7.76	24.54

5 rows × 32 columns

```
df.shape
```

```
(569, 32)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave_points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
```

```
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave_points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave_points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

```
df.describe()
```

	id	radius_mean	texture_mean	pe
count	5.690000e+02	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	
std	1.250206e+08	3.524049	4.301036	
min	8.670000e+03	6.981000	9.710000	
25%	8.692180e+05	11.700000	16.170000	
50%	9.060240e+05	13.370000	18.840000	
75%	8.813129e+06	15.780000	21.800000	
max	9.113205e+08	28.110000	39.280000	

8 rows × 31 columns

```
df.isnull().values.any()
```

```
False
```

```
df.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave_points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
```

```
concavity_se               0
concave_points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave_points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

```
df
```

	id	diagnosis	radius_mean	texture_mea
0	842302	M	17.99	10.3
1	842517	M	20.57	17.7
2	84300903	M	19.69	21.2
3	84348301	M	11.42	20.3
4	84358402	M	20.29	14.3
...	...	...	...	
564	926424	M	21.56	22.3
565	926682	M	20.13	28.2
566	926954	M	16.60	28.0
567	927241	M	20.60	29.3
568	92751	B	7.76	24.5

569 rows × 32 columns

```
df.drop(columns="id",axis=1,inplace=True)
df
```

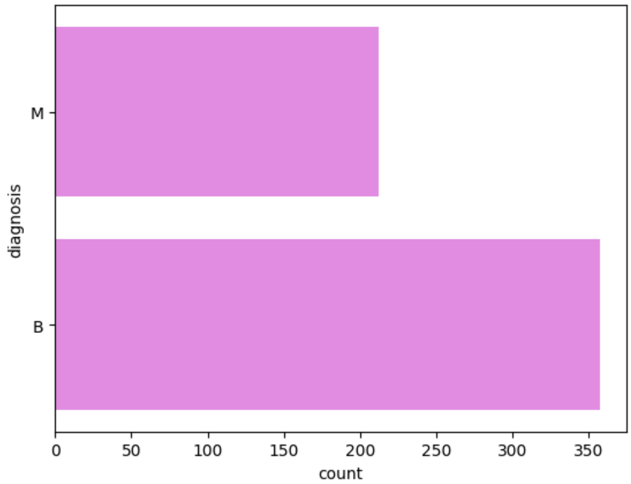
	diagnosis	radius_mean	texture_mean	perimet
0	M	17.99	10.38	
1	M	20.57	17.77	
2	M	19.69	21.25	
3	M	11.42	20.38	
4	M	20.29	14.34	
...	...	...	...	
564	M	21.56	22.39	
565	M	20.13	28.25	
566	M	16.60	28.08	
567	M	20.60	29.33	
568	B	7.76	24.54	

569 rows × 31 columns

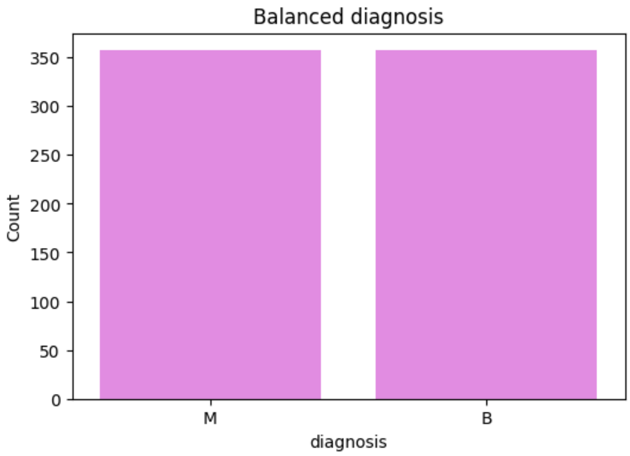
```
X = df.drop('diagnosis', axis = 1)
y = df['diagnosis']
```

```
sns.countplot(y,color='violet')
```

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



```
import seaborn as sns
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler

# Assuming you have already imported and preprocessed your data

# Upsample the minority class (class 1)
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(df.drop('diagnosis', a

# Create a DataFrame with the upsampled data
df_balanced = pd.DataFrame(X_resampled, columns=df.drop('diagnosis
df_balanced['diagnosis'] = y_resampled

# Plot the count of balanced diagnosis after balancing the dataset
plt.figure(figsize=(6, 4))
sns.countplot(x='diagnosis', color='violet',data=df_balanced)
plt.title('Balanced diagnosis')
plt.xlabel('diagnosis')
plt.ylabel('Count')
plt.show()
```




```
from sklearn.preprocessing import LabelEncoder
columns_to_encode = ['diagnosis']
label_encoder = LabelEncoder()
for column in columns_to_encode:
  df[column]=label_encoder.fit_transform(df[column])
cols=df.columns.to_list()
```

```
df
```

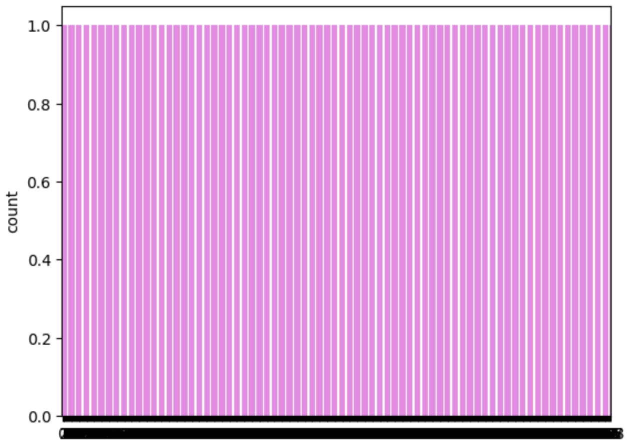
	diagnosis	radius_mean	texture_mean	perimet
0	1	17.99	10.38	
1	1	20.57	17.77	
2	1	19.69	21.25	
3	1	11.42	20.38	
4	1	20.29	14.34	
...	...	...	...	
564	1	21.56	22.39	
565	1	20.13	28.25	
566	1	16.60	28.08	
567	1	20.60	29.33	
568	0	7.76	24.54	

569 rows × 31 columns

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

```
sns.countplot(df['diagnosis'],color='violet')
```

```
<Axes: ylabel='count'>
```

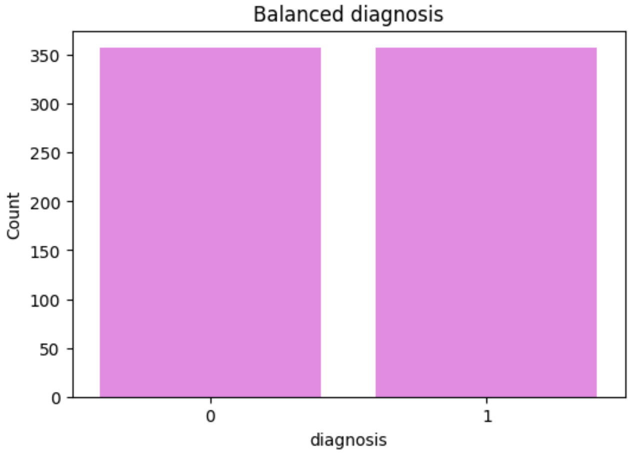



```
df
```

	diagnosis	radius_mean	texture_mean	perimet
0	1	17.99	10.38	
1	1	20.57	17.77	
2	1	19.69	21.25	
3	1	11.42	20.38	
4	1	20.29	14.34	
...	...	...	...	
564	1	21.56	22.39	
565	1	20.13	28.25	
566	1	16.60	28.08	
567	1	20.60	29.33	
568	0	7.76	24.54	

569 rows × 31 columns

```
import seaborn as sns
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler

# Assuming you have already imported and preprocessed your data

# Upsample the minority class (class 1)
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(df.drop('diagnosis', a

# Create a DataFrame with the upsampled data
df_balanced = pd.DataFrame(X_resampled, columns=df.drop('diagnosis
df_balanced['diagnosis'] = y_resampled

# Plot the count of balanced diagnosis after balancing the dataset
plt.figure(figsize=(6, 4))
sns.countplot(x='diagnosis', color='violet',data=df_balanced)
plt.title('Balanced diagnosis')
plt.xlabel('diagnosis')
plt.ylabel('Count')
plt.show()
```




```
# Scaling the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

```
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]])
```

```
pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/loc
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/lo
```

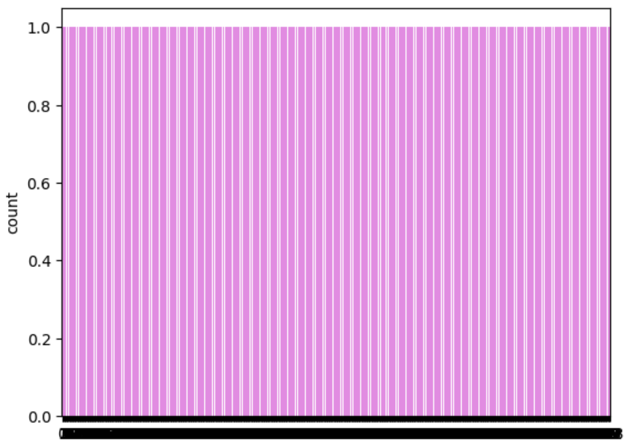
```
df
```

	diagnosis	radius_mean	texture_mean	perimet
0	1	17.99	10.38	
1	1	20.57	17.77	
2	1	19.69	21.25	
3	1	11.42	20.38	
4	1	20.29	14.34	
...	...	...	...	
564	1	21.56	22.39	
565	1	20.13	28.25	
566	1	16.60	28.08	
567	1	20.60	29.33	
568	0	7.76	24.54	

569 rows × 31 columns

```
# Oversampling
from imblearn.over_sampling import RandomOverSampler

# Oversample the minority class
oversampler = RandomOverSampler()
X_resampled, y_resampled = oversampler.fit_resample(X_scaled, y)
```

```
# Distribution of the target variable 'diagnonis' after oversampli
sns.countplot(y_resampled,color='violet');
```




```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_sc = sc.fit_transform(X_train)
X_test_sc = sc.transform(X_test)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(X_train,y_train)
y_pred_logreg=logreg.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
print(confusion_matrix(y_test,y_pred_logreg))
print('Accuracy:',accuracy_score(y_test,y_pred_logreg))
print('Precision:',precision_score(y_test,y_pred_logreg,pos_label=
print('Recall:',recall_score(y_test,y_pred_logreg,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_logreg,pos_label=1))
print(classification_report(y_test, y_pred_logreg))
```

```
[[63  4]
 [ 2 45]]
Accuracy: 0.9473684210526315
Precision: 0.9183673469387755
Recall: 0.9574468085106383
F1 score: 0.9375000000000001
              precision    recall  f1-score   support

           0       0.97      0.94      0.95        67
           1       0.92      0.96      0.94        47

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.95       114
weighted avg       0.95      0.95      0.95       114

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7)# default minkwoski
knn.fit(X_train,y_train)
y_pred_knn = knn.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
print(confusion_matrix(y_test,y_pred_knn))
print('Accuracy:',accuracy_score(y_test,y_pred_knn))
print('Precision:',precision_score(y_test,y_pred_knn,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_knn,pos_label=1))
print('f1 score:',f1_score(y_test,y_pred_knn,pos_label=1))
print(classification_report(y_test, y_pred_knn))
```

```
[[64  3]
 [ 3 44]]
Accuracy: 0.9473684210526315
Precision: 0.9361702127659575
Recall: 0.9361702127659575
f1 score: 0.9361702127659575
              precision    recall  f1-score   support
```

```
              0       0.96      0.96      0.96        67
              1       0.94      0.94      0.94        47

       accuracy                           0.95       114
      macro avg       0.95      0.95      0.95       114
   weighted avg       0.95      0.95      0.95       114
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7,metric='euclidean')
knn.fit(X_train,y_train)
y_pred_knn=knn.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
print(confusion_matrix(y_test,y_pred_knn))
print('Accuracy:',accuracy_score(y_test,y_pred_knn))
print('Precision:',precision_score(y_test,y_pred_knn,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_knn,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_knn,pos_label=1))
print(classification_report(y_test, y_pred_knn))
```

```
[[64  3]
 [ 3 44]]
Accuracy: 0.9473684210526315
Precision: 0.9361702127659575
Recall: 0.9361702127659575
F1 score: 0.9361702127659575
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        67
           1       0.94      0.94      0.94        47

    accuracy                           0.95       114
   macro avg       0.95      0.95      0.95       114
weighted avg       0.95      0.95      0.95       114
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7,metric='manhattan')
knn.fit(X_train,y_train)
y_pred_knn=knn.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
print(confusion_matrix(y_test,y_pred_knn))
print('Accuracy:',accuracy_score(y_test,y_pred_knn))
print('Precision:',precision_score(y_test,y_pred_knn,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_knn,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_knn,pos_label=1))
print(classification_report(y_test, y_pred_knn))
```

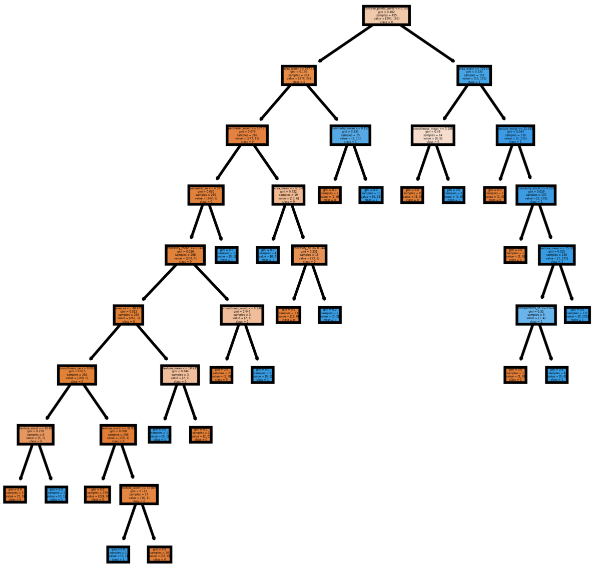
```
[[65  2]
 [ 3 44]]
Accuracy: 0.956140350877193
Precision: 0.9565217391304348
Recall: 0.9361702127659575
F1 score: 0.9462365591397849
              precision    recall  f1-score   support

           0       0.96      0.97      0.96        67
           1       0.96      0.94      0.95        47

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier(random_state=1)
dtree.fit(X_train,y_train)
y_pred_dt=dtree.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
print(confusion_matrix(y_test,y_pred_dt))
print('Accuracy:',accuracy_score(y_test,y_pred_dt))
print('Precision:',precision_score(y_test,y_pred_dt,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_dt,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_dt,pos_label=1))
print(classification_report(y_test, y_pred_dt))
```

```
[[60  7]
 [ 4 43]]
Accuracy: 0.9035087719298246
Precision: 0.86
Recall: 0.9148936170212766
F1 score: 0.8865979381443299
              precision    recall  f1-score   support

           0       0.94      0.90      0.92        67
           1       0.86      0.91      0.89        47

    accuracy                           0.90       114
   macro avg       0.90      0.91      0.90       114
weighted avg       0.91      0.90      0.90       114
```

```
from sklearn.tree import plot_tree
fn=list(X_train)
cn=['0','1']
plt.figure(figsize=(4,4),dpi=600)
plot_tree(dtree,feature_names=fn,class_names=cn,filled=True);
```

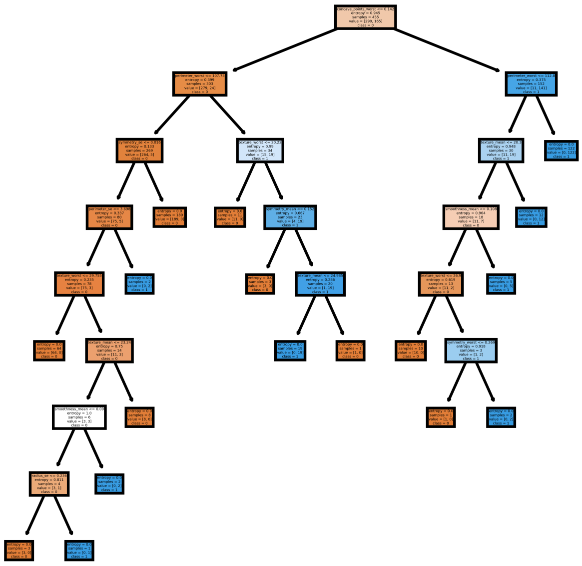


```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier(criterion='entropy')
dtree.fit(X_train,y_train)
y_pred_dt=dtree.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
print(confusion_matrix(y_test,y_pred_dt))
print('Accuracy:',accuracy_score(y_test,y_pred_dt))
print('Precision:',precision_score(y_test,y_pred_dt,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_dt,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_dt,pos_label=1))
print(classification_report(y_test, y_pred_dt))
```

```
[[61  6]
 [ 4 43]]
Accuracy: 0.9122807017543859
Precision: 0.8775510204081632
Recall: 0.9148936170212766
F1 score: 0.8958333333333333
              precision    recall  f1-score   support
```

```
              0       0.94      0.91      0.92        67
              1       0.88      0.91      0.90        47

       accuracy                           0.91       114
      macro avg       0.91      0.91      0.91       114
   weighted avg       0.91      0.91      0.91       114
```

```
#Visualizing the tree
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
fn=list(X_train)
cn=['0','1']
plt.figure(figsize=(4,4),dpi=1000)
plot_tree(dtree,feature_names=fn,class_names=cn,filled=True);
```



SVM-Support Vector Machine

```
from sklearn import svm
clf=svm.SVC(kernel='linear',C=0.01)
clf.fit(X_train,y_train)
y_pred_svm=clf.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score,precis
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_svm))
print(accuracy_score(y_test,y_pred_svm))
print('Precision:',precision_score(y_test,y_pred_svm,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_svm,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_svm,pos_label=1))
print(classification_report(y_test,y_pred_svm))
```

```
[[62  5]
 [ 3 44]]
0.9298245614035088
Precision: 0.8979591836734694
Recall: 0.9361702127659575
F1 score: 0.9166666666666666
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        67
           1       0.90      0.94      0.92        47

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114
```

Random Forest-Ensemble Model

```
from sklearn.ensemble import RandomForestClassifier
rf_classifier=RandomForestClassifier(n_estimators=10)
rf_classifier.fit(X_train,y_train)
y_pred_rf=rf_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_rf))
print(accuracy_score(y_test,y_pred_rf))
print('Precision:',precision_score(y_test,y_pred_rf,pos_label=1))
print('Recall:',recall_score(y_test,y_pred_rf,pos_label=1))
print('F1 score:',f1_score(y_test,y_pred_rf,pos_label=1))
print(classification_report(y_test,y_pred_rf))
```

```
[[66  1]
 [ 3 44]]
0.9649122807017544
Precision: 0.9777777777777777
Recall: 0.9361702127659575
F1 score: 0.9565217391304347
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        67
           1       0.98      0.94      0.96        47

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

Boosting:

Gradient Boosting

AdaBoost

XGBoost

Catboost

```
from sklearn.ensemble import GradientBoostingClassifier
gradient_booster=GradientBoostingClassifier(learning_rate=0.1)
gradient_booster.fit(X_train,y_train)
y_pred_gradboost=gradient_booster.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_gradboost))
print(accuracy_score(y_test,y_pred_gradboost))
print(classification_report(y_test,y_pred_gradboost))
```

```
[[65  2]
 [ 2 45]]
0.9649122807017544
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        67
           1       0.96      0.96      0.96        47

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114
```

```
from sklearn.ensemble import AdaBoostClassifier
abc=AdaBoostClassifier(learning_rate=0.1)
abc.fit(X_train,y_train)
y_pred_abc=abc.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_abc))
print(accuracy_score(y_test,y_pred_abc))
print(classification_report(y_test,y_pred_abc))
```

```
[[65  2]
 [ 3 44]]
0.956140350877193
              precision    recall  f1-score   support

           0       0.96      0.97      0.96        67
           1       0.96      0.94      0.95        47

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/pytho
Requirement already satisfied: numpy in /usr/local/lib/python3
Requirement already satisfied: scipy in /usr/local/lib/python3
```

```
from xgboost import XGBClassifier
#print(y_train.unique())
#print(y_test.unique())
#y_train = y_train.replace({'B': 0, 'M': 1})
#y_test = y_test.replace({'B': 0, 'M': 1})
model=XGBClassifier(learning_rate=1)
model.fit(X_train,y_train)
y_pred_xgb=model.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_xgb))
print(accuracy_score(y_test,y_pred_xgb))
print(classification_report(y_test,y_pred_xgb))
```

```
[[67  0]
 [ 2 45]]
0.9824561403508771
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        67
           1       1.00      0.96      0.98        47

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

```
pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.3-cp310-cp310-manylinux2014_x86_64.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.5/98.5 MB 3.3
Requirement already satisfied: graphviz in /usr/local/lib/pyth
Requirement already satisfied: matplotlib in /usr/local/lib/py
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/
Requirement already satisfied: scipy in /usr/local/lib/python3
Requirement already satisfied: plotly in /usr/local/lib/python
Requirement already satisfied: six in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/
Requirement already satisfied: fonttools>=4.22.0 in /usr/local
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local
Requirement already satisfied: packaging>=20.0 in /usr/local/l
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/l
Installing collected packages: catboost
Successfully installed catboost-1.2.3
```

```
from catboost import CatBoostClassifier
model1=CatBoostClassifier()
model1.fit(X_train,y_train)
y_pred_cat=model1.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_cat))
print(accuracy_score(y_test,y_pred_cat))
print(classification_report(y_test,y_pred_cat))
```

```
962:    learn: 0.0118960        total: 8.68s    remaining:
963:    learn: 0.0118732        total: 8.69s    remaining:
964:    learn: 0.0118458        total: 8.7s     remaining:
965:    learn: 0.0118232        total: 8.7s     remaining:
966:    learn: 0.0117976        total: 8.71s    remaining:
967:    learn: 0.0117850        total: 8.72s    remaining:
968:    learn: 0.0117577        total: 8.73s    remaining:
969:    learn: 0.0117296        total: 8.74s    remaining:
970:    learn: 0.0117062        total: 8.75s    remaining:
971:    learn: 0.0116965        total: 8.76s    remaining:
972:    learn: 0.0116789        total: 8.77s    remaining:
973:    learn: 0.0116427        total: 8.78s    remaining:
974:    learn: 0.0116143        total: 8.79s    remaining:
975:    learn: 0.0116113        total: 8.79s    remaining:
976:    learn: 0.0116029        total: 8.8s     remaining:
977:    learn: 0.0115812        total: 8.81s    remaining:
978:    learn: 0.0115531        total: 8.82s    remaining:
979:    learn: 0.0115317        total: 8.83s    remaining:
980:    learn: 0.0115307        total: 8.84s    remaining:
981:    learn: 0.0115189        total: 8.85s    remaining:
982:    learn: 0.0114960        total: 8.85s    remaining:
983:    learn: 0.0114938        total: 8.86s    remaining:
984:    learn: 0.0114851        total: 8.87s    remaining:
985:    learn: 0.0114655        total: 8.88s    remaining:
986:    learn: 0.0114497        total: 8.88s    remaining:
987:    learn: 0.0114241        total: 8.89s    remaining:
988:    learn: 0.0114160        total: 8.9s     remaining:
989:    learn: 0.0114035        total: 8.91s    remaining:
990:    learn: 0.0113811        total: 8.92s    remaining:
991:    learn: 0.0113660        total: 8.93s    remaining:
992:    learn: 0.0113450        total: 8.93s    remaining:
993:    learn: 0.0113286        total: 8.94s    remaining:
994:    learn: 0.0113075        total: 8.95s    remaining:
995:    learn: 0.0112741        total: 8.96s    remaining:
996:    learn: 0.0112649        total: 8.98s    remaining:
997:    learn: 0.0112396        total: 8.98s    remaining:
998:    learn: 0.0112154        total: 8.99s    remaining:
999:    learn: 0.0112034        total: 9s       remaining:
[[66  1]
 [ 2 45]]
0.9736842105263158
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        67
           1       0.98      0.96      0.97        47

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

```
print("y_pred_logreg",y_pred_logreg.shape)
print("y_pred_knn",y_pred_knn.shape)
print("y_pred_svm",y_pred_svm.shape)
print("y_pred_dt",y_pred_dt.shape)
```

```
y_pred_logreg (114,)
y_pred_knn (114,)
y_pred_svm (114,)
y_pred_dt (114,)
```

```
print(X[:5])
```

```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoot
0        17.99         10.38          122.80     1001.0
1        20.57         17.77          132.90     1326.0
2        19.69         21.25          130.00     1203.0
3        11.42         20.38           77.58      386.1
4        20.29         14.34          135.10     1297.0
```

```
   compactness_mean  concavity_mean  concave_points_mean  symm
0           0.27760          0.3001              0.14710
1           0.07864          0.0869              0.07017
2           0.15990          0.1974              0.12790
3           0.28390          0.2414              0.10520
4           0.13280          0.1980              0.10430

   fractal_dimension_mean  ...  radius_worst  texture_worst  p
0                 0.07871  ...         25.38          17.33
1                 0.05667  ...         24.99          23.41
2                 0.05999  ...         23.57          25.53
3                 0.09744  ...         14.91          26.50
4                 0.05883  ...         22.54          16.67

   area_worst  smoothness_worst  compactness_worst  concavity_
0      2019.0            0.1622             0.6656           0
1      1956.0            0.1238             0.1866           0
2      1709.0            0.1444             0.4245           0
3       567.7            0.2098             0.8663           0
4      1575.0            0.1374             0.2050           0

   concave_points_worst  symmetry_worst  fractal_dimension_wor
0                0.2654          0.4601                0.118
1                0.1860          0.2750                0.089
2                0.2430          0.3613                0.087
3                0.2575          0.6638                0.173
4                0.1625          0.2364                0.076

[5 rows x 30 columns]
```

Ensemble

```
from sklearn.ensemble import RandomForestClassifier
X=np.array([y_pred_logreg,y_pred_knn,y_pred_svm,y_pred_dt]).T
meta_learner=RandomForestClassifier()
meta_learner2=meta_learner.fit(X,y_test)
ensemble_prediction=meta_learner.predict(X)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,ensemble_prediction))
print(accuracy_score(y_test,ensemble_prediction))
print(classification_report(y_test,ensemble_prediction))
```

```
[[65  2]
 [ 2 45]]
0.9649122807017544
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        67
           1       0.96      0.96      0.96        47

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114
```

```
from sklearn.ensemble import RandomForestClassifier
X=np.array([y_pred_cat,y_pred_xgb,y_pred_abc,y_pred_gradboost,y_pr
meta_learner=RandomForestClassifier()
meta_learner2=meta_learner.fit(X,y_test)
ensemble_prediction=meta_learner.predict(X)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,ensemble_prediction))
print(accuracy_score(y_test,ensemble_prediction))
print(classification_report(y_test,ensemble_prediction))
```

```
[[67  0]
 [ 0 47]]
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        67
           1       1.00      1.00      1.00        47

    accuracy                           1.00       114
   macro avg       1.00      1.00      1.00       114
weighted avg       1.00      1.00      1.00       114
```

```
from sklearn.ensemble import RandomForestClassifier
X=np.array([y_pred_logreg,y_pred_svm,y_pred_xgb,y_pred_abc,y_pred_
meta_learner=RandomForestClassifier()
meta_learner.fit(X,y_test)
ensemble_prediction=meta_learner.predict(X)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,ensemble_prediction))
print(accuracy_score(y_test,ensemble_prediction))
print(classification_report(y_test,ensemble_prediction))
```

```
[[67  0]
 [ 2 45]]
0.9824561403508771
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        67
           1       1.00      0.96      0.98        47

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

```
!pip install lime
import lime
import lime.lime_tabular
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 4
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/py
Requirement already satisfied: numpy in /usr/local/lib/python3
Requirement already satisfied: scipy in /usr/local/lib/python3
Requirement already satisfied: tqdm in /usr/local/lib/python3.
Requirement already satisfied: scikit-learn>=0.18 in /usr/loca
Requirement already satisfied: scikit-image>=0.12 in /usr/loca
Requirement already satisfied: networkx>=2.2 in /usr/local/lib
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>
Requirement already satisfied: imageio>=2.4.1 in /usr/local/li
Requirement already satisfied: tifffile>=2019.7.26 in /usr/loc
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local
Requirement already satisfied: packaging>=20.0 in /usr/local/l
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib
```

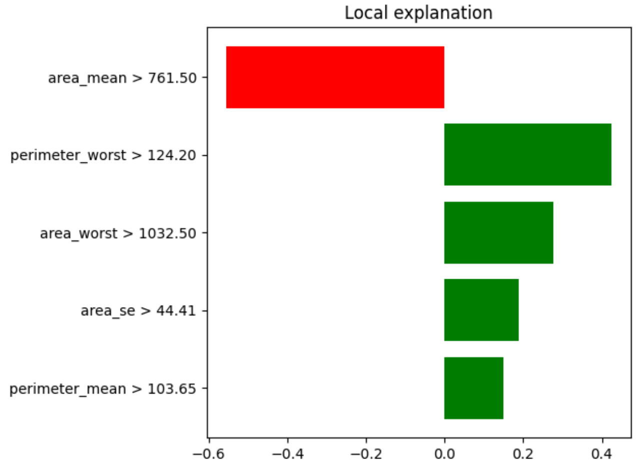
```
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/lc
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/
Requirement already satisfied: fonttools>=4.22.0 in /usr/local
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/
Requirement already satisfied: python-dateutil>=2.7 in /usr/lc
Requirement already satisfied: six>=1.5 in /usr/local/lib/pyth
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.w
  Stored in directory: /root/.cache/pip/wheels/fd/a2/af/9ac0a1
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

```
from sklearn import svm
clf=svm.SVC(kernel='linear',C=0.01)
clf.fit(X_train,y_train)
y_pred_svm=clf.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test,y_pred_svm))
print(accuracy_score(y_test,y_pred_svm))
print(classification_report(y_test,y_pred_svm))
```

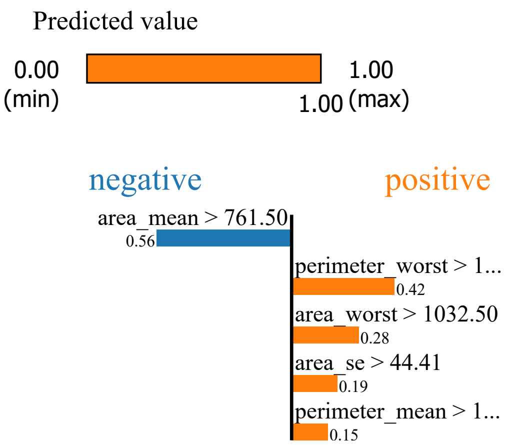
```
[[62  5]
 [ 3 44]]
0.9298245614035088
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        67
           1       0.90      0.94      0.92        47

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114

[[62  5]
 [ 3 44]]
0.9298245614035088
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        67
           1       0.90      0.94      0.92        47

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114
```

```
from lime.lime_tabular import LimeTabularExplainer
explainer=LimeTabularExplainer(X_train.values,feature_names=X_trai
exp=explainer.explain_instance(X_train.values[100],clf.predict,num
exp.as_pyplot_figure()
from matplotlib import pyplot as plt
plt.tight_layout()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ba
  warnings.warn(
```




```
exp.show_in_notebook(show_table=True)
```

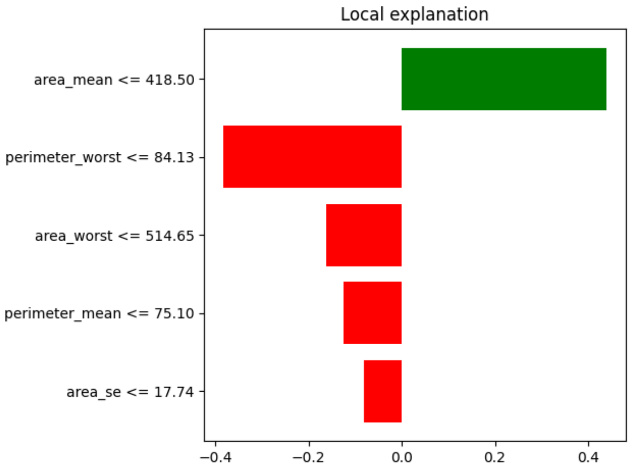


Feature	Value
area_mean	1682.00

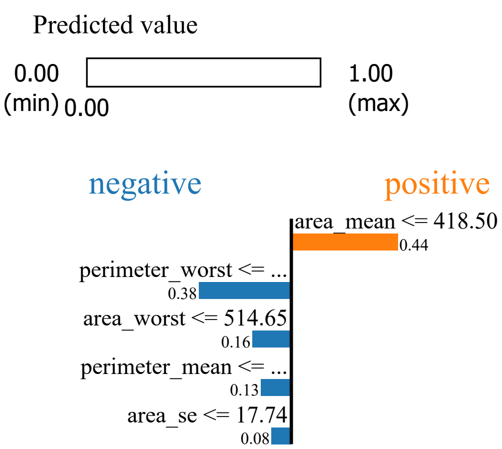
Double-click (or enter) to edit

```
from lime.lime_tabular import LimeTabularExplainer
explainer=LimeTabularExplainer(X_train.values,feature_names=X_trai
exp=explainer.explain_instance(X_train.values[50],clf.predict,num_
exp.as_pyplot_figure()
from matplotlib import pyplot as plt
plt.tight_layout()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ba

X does not have valid feature names, but SVC was f
```



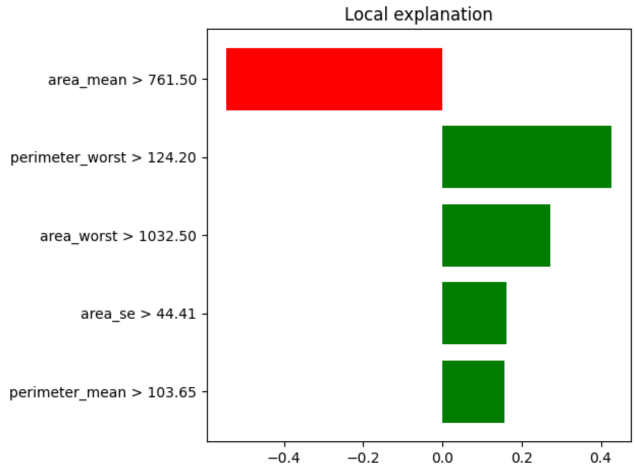

```
exp.show_in_notebook(show_table=True)
```



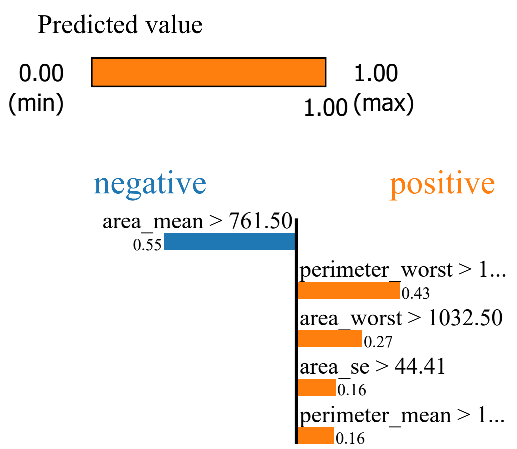
Feature	Value
area_mean	399.80

```
from lime.lime_tabular import LimeTabularExplainer
explainer=LimeTabularExplainer(X_train.values,feature_names=X_trai
exp=explainer.explain_instance(X_train.values[100],clf.predict,num
exp.as_pyplot_figure()
from matplotlib import pyplot as plt
plt.tight_layout()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ba

X does not have valid feature names, but SVC was f
```



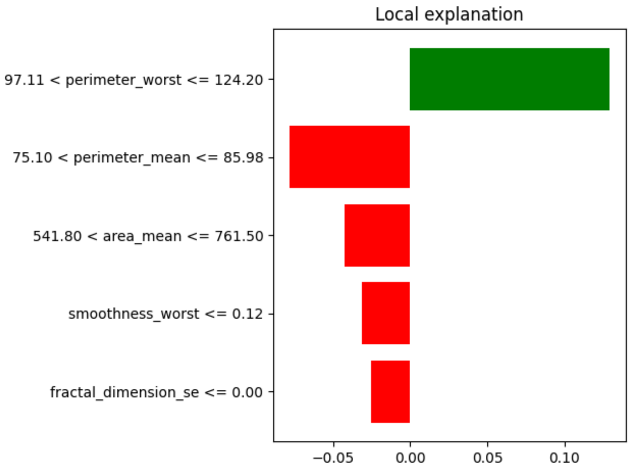

```
exp.show_in_notebook(show_table=True)
```



Feature	Value
area_mean	1682.00

```
from lime.lime_tabular import LimeTabularExplainer
explainer=LimeTabularExplainer(X_train.values,feature_names=X_train
exp=explainer.explain_instance(X_train.values[125],clf.predict,num_
exp.as_pyplot_figure()
from matplotlib import pyplot as plt
plt.tight_layout()
```

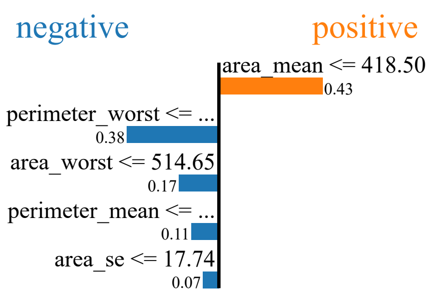
```
/usr/local/lib/python3.10/dist-packages/sklearn/ba

X does not have valid feature names, but SVC was f
```




```
exp.show_in_notebook(show_table=True)
```

Predicted value

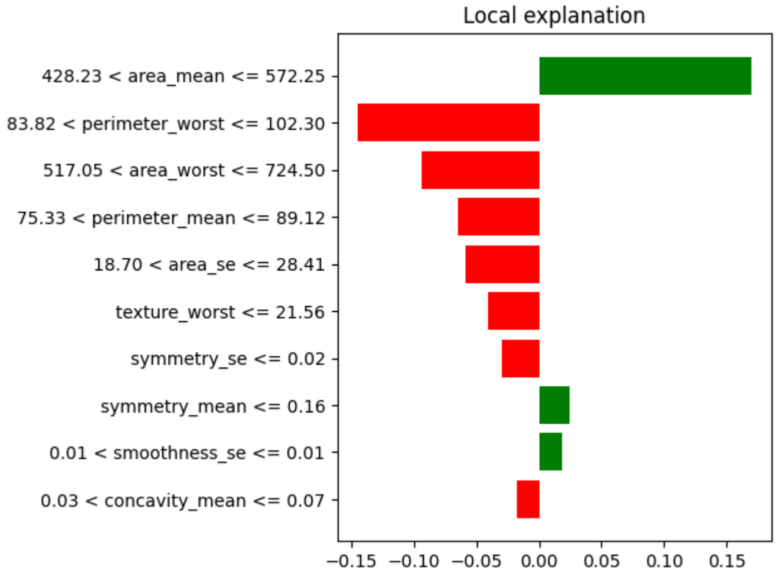
0.00 (min) 0.00 1.00 (max)



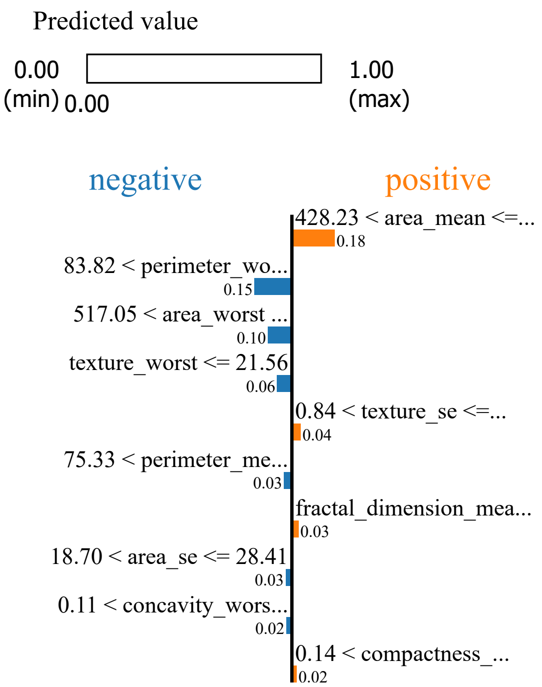
Feature	Value
area_mean	355.30

```
from lime.lime_tabular import LimeTabularExplainer
explainer=LimeTabularExplainer(X_test.values,feature_names=X_test.c
exp=explainer.explain_instance(X_test.values[25],clf.predict,num_fe
exp.as_pyplot_figure()
from matplotlib import pyplot as plt
plt.tight_layout()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: U
  warnings.warn(
```




```
exp.show_in_notebook(show_table=True)
```



Feature	Value
area_mean	442.70
perimeter_worst	87.64
area_worst	589.50
texture_worst	20.14
texture_se	1.04
perimeter_mean	76.14
fractal_dimension_mean	0.06

```
pip install shapash
```

```
Requirement already satisfied: Flask<2.3.0 in /usr/local/li
Requirement already satisfied: dash>=2.3.1 in /usr/local/li
Requirement already satisfied: dash-bootstrap-components>=1
Requirement already satisfied: dash-core-components>=2.0.0
Requirement already satisfied: dash-daq>=0.5.0 in /usr/loca
Requirement already satisfied: dash-html-components>=2.0.0
Requirement already satisfied: dash-renderer==1.8.3 in /usr
Requirement already satisfied: dash-table>=5.0.0 in /usr/lo
Requirement already satisfied: nbformat>4.2.0 in /usr/local
Requirement already satisfied: numba>=0.53.1 in /usr/local/
Requirement already satisfied: scikit-learn<1.4,>=1.0.1 in
Requirement already satisfied: category-encoders>=2.6.0 in
Requirement already satisfied: scipy>=0.19.1 in /usr/local/
Requirement already satisfied: statsmodels>=0.9.0 in /usr/l
Requirement already satisfied: patsy>=0.5.1 in /usr/local/l
Requirement already satisfied: Werkzeug<3.1 in /usr/local/l
Requirement already satisfied: importlib-metadata in /usr/l
Requirement already satisfied: typing-extensions>=4.1.1 in
Requirement already satisfied: requests in /usr/local/lib/p
Requirement already satisfied: retrying in /usr/local/lib/p
Requirement already satisfied: nest-asyncio in /usr/local/l
Requirement already satisfied: setuptools in /usr/local/lib
Requirement already satisfied: Jinja2>=3.0 in /usr/local/li
Requirement already satisfied: itsdangerous>=2.0 in /usr/lo
Requirement already satisfied: click>=8.0 in /usr/local/lib
Requirement already satisfied: contourpy>=1.0.1 in /usr/loc
Requirement already satisfied: cycler>=0.10 in /usr/local/l
Requirement already satisfied: fonttools>=4.22.0 in /usr/lo
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/lo
Requirement already satisfied: packaging>=20.0 in /usr/loca
Requirement already satisfied: pillow>=6.2.0 in /usr/local/
Requirement already satisfied: pyparsing>=2.3.1 in /usr/loc
Requirement already satisfied: python-dateutil>=2.7 in /usr
Requirement already satisfied: fastjsonschema in /usr/local
Requirement already satisfied: jsonschema>=2.6 in /usr/loca
Requirement already satisfied: jupyter-core in /usr/local/l
Requirement already satisfied: traitlets>=5.1 in /usr/local
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 i
Requirement already satisfied: pytz>=2020.1 in /usr/local/l
Requirement already satisfied: tenacity>=6.2.0 in /usr/loca
Requirement already satisfied: joblib>=1.1.1 in /usr/local/
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/l
Requirement already satisfied: slicer==0.0.7 in /usr/local/
Requirement already satisfied: cloudpickle in /usr/local/li
Requirement already satisfied: MarkupSafe>=2.0 in /usr/loca
Requirement already satisfied: attrs>=22.2.0 in /usr/local/
Requirement already satisfied: jsonschema-specifications>=2
Requirement already satisfied: referencing>=0.28.4 in /usr/
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local
Requirement already satisfied: six in /usr/local/lib/python
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/
Requirement already satisfied: platformdirs>=2.5 in /usr/lo
Requirement already satisfied: charset-normalizer<4,>=2 in
Requirement already satisfied: idna<4,>=2.5 in /usr/local/l
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/l
Requirement already satisfied: certifi>=2017.4.17 in /usr/l
```

```
model=RandomForestClassifier(max_depth=5,random_state=42,n_estimat
model2=model.fit(X_train,y_train)
rf_y_pred=model2.predict(X_test)
rf_y_pred
```

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1,
1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
0, 0, 1, 0,
```

```
       1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
1, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 1, 1,
       0, 1, 1, 0])
```

```
fi=pd.DataFrame({'Feature':X_train.columns,'Importance':model2.fea
fi.sort_values(by='Importance',ascending=False,ignore_index=True)
```

	Feature	Importance
0	concave_points_worst	0.178218
1	radius_mean	0.108297
2	concave_points_mean	0.097164
3	radius_worst	0.094976
4	perimeter_worst	0.091844
5	area_worst	0.090185
6	concavity_worst	0.066257
7	area_se	0.064503
8	perimeter_mean	0.050140
9	compactness_worst	0.034604
10	symmetry_worst	0.025056
11	texture_worst	0.013426
12	texture_se	0.011878
13	area_mean	0.011625
14	fractal_dimension_worst	0.011422
15	perimeter_se	0.007925
16	concavity_se	0.007664
17	texture_mean	0.007589
18	radius_se	0.004198
19	smoothness_mean	0.003778
20	symmetry_mean	0.003546
21	fractal_dimension_mean	0.003253
22	smoothness_worst	0.003031
23	fractal_dimension_se	0.002518
24	symmetry_se	0.002296
25	smoothness_se	0.001429
26	compactness_se	0.001406
27	compactness_mean	0.001284
28	concavity_mean	0.000490
29	concave_points_se	0.000000

```
from shapash.explainer.smart_explainer import SmartExplainer
```

Disk 82.85 GB availat