```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split , cross_val_score  , GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score , accuracy_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
data = pd.read_csv('data.csv')
```

```
data.describe()
```

Show hidden output

```
data.drop(['id', 'Unnamed: 32'], axis = 1, inplace = True)
```

```
data.head()
```

```
data['diagnosis'].unique()
```

```
array(['M', 'B'], dtype=object)
```

```
data['diagnosis'] = data['diagnosis'].apply(lambda val: 1 if val == 'M' else 0)
```

```
data.head()
```

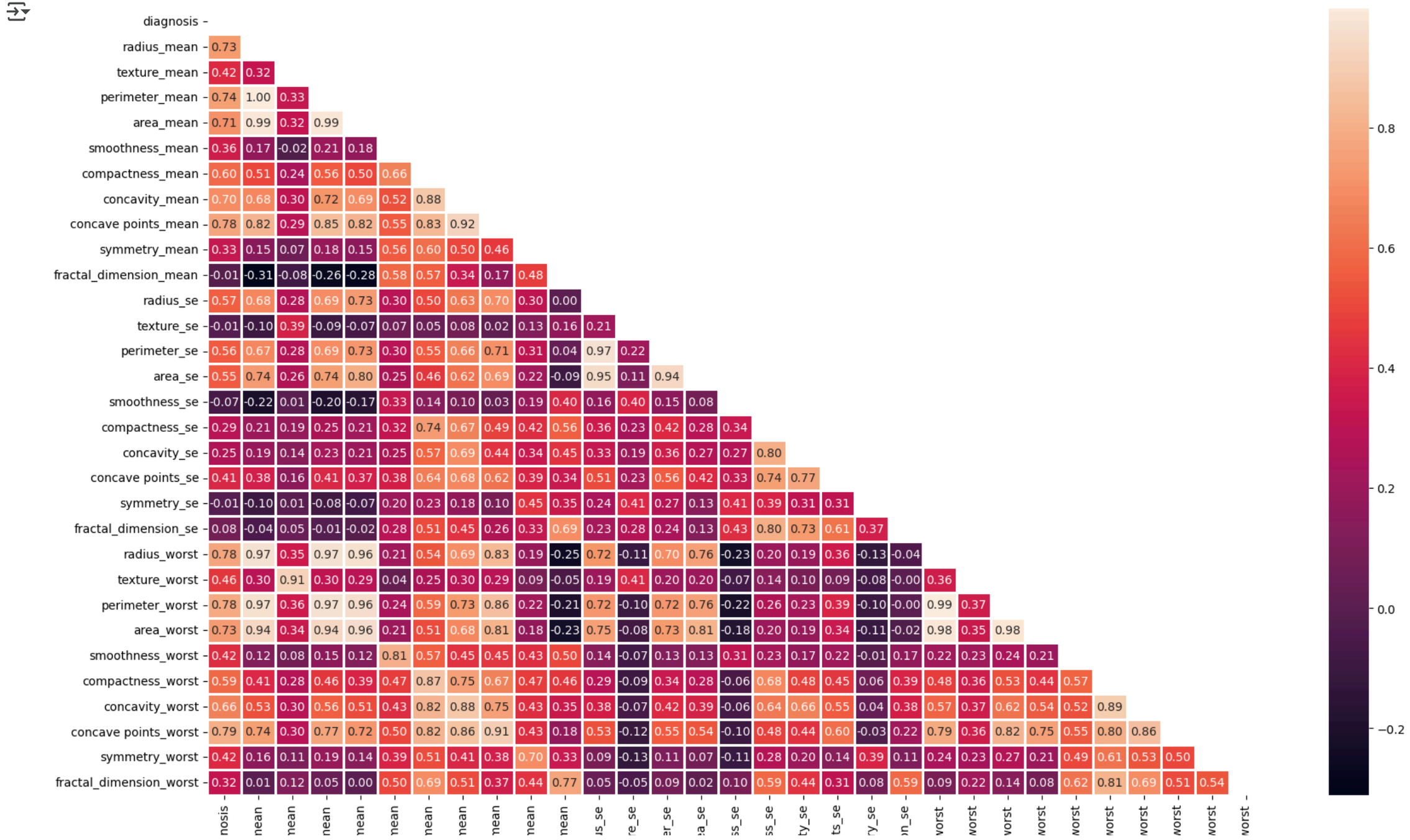
```
data.isnull().sum()
```

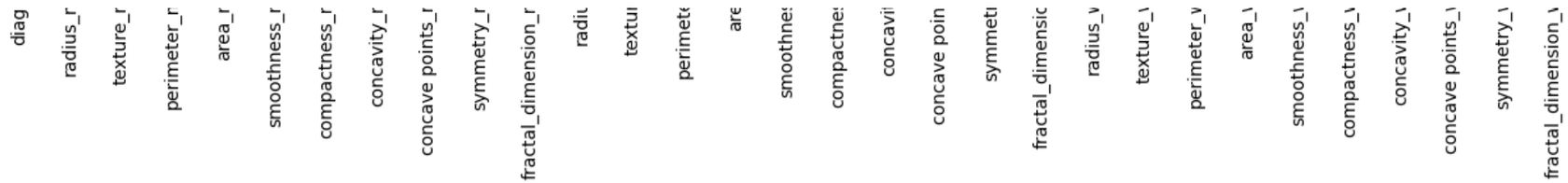
```
data.dtypes
```

```
fig, ax = plt.subplots(16,2, figsize=(12,12))
axes_ = [axes_row for axes in ax for axes_row in axes]

for i, j in enumerate(data.columns):
    g = sns.boxplot(x = data[j], ax = axes_[i])
    g.set_title(j)
    plt.tight_layout()
```

```
plt.figure(figsize = (20, 15))
plotnumber = 1

for column in data:
    if plotnumber <= 30:
        ax = plt.subplot(5, 6, plotnumber)
        sns.distplot(data[column])
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

```
plt.figure(figsize = (20, 12))

corr = data.corr()
mask = np.triu(np.ones_like(corr, dtype = bool))

sns.heatmap(corr, mask = mask, linewidths = 1, annot = True, fmt = ".2f")
plt.show()
```


diagnosis
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst
0.8
0.6
0.4
0.2
0.0
−0.2




```
corr_matrix = data.corr().abs()

mask = np.triu(np.ones_like(corr_matrix, dtype = bool))
tri_data = corr_matrix.mask(mask)

to_drop = [x for x in tri_data.columns if any(tri_data[x] > 0.92)]

data = data.drop(to_drop, axis = 1)

print(f"The reduced dataframe has {df.shape[1]} columns.")
```

```
The reduced dataframe has 23 columns.
```

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

```
X.shape
```

```
(569, 22)
```

```
base = DecisionTreeClassifier()
```

```

```

```
model = AdaBoostClassifier(estimator=base)
```

```
param_grid = {
    'n_estimators': [50, 100, 180 , 200],
    'learning_rate': [0.01, 0.1, 0.5, 1.0],

}


grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
                           cv=5, scoring='accuracy', n_jobs=-1)
```

```
grid_search.fit(X, y)


print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'learning_rate': 0.1, 'n_estimators': 200}
```

```
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size=0.3 , random_state=0)
```

```
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

```
ada = AdaBoostClassifier(base, n_estimators = 180)
ada.fit(X_train, y_train)
```

```
AdaBoostClassifier
```

```
y_pred = ada.predict(X_test)
```

```
print(accuracy_score(y_test, y_pred))
```

```
0.9298245614035088
```

```
print(accuracy_score(y_train, ada.predict(X_train)))

ada_acc = accuracy_score(y_test, y_pred)
print(ada_acc)
```

```
1.0
0.9298245614035088
```