ابتدا مجموعه داده را وارد می‌کنیم.

برای هر ویژگی(ستون) با استفاده از کد پایین، هیستوگرام ها را رسم میکنیم:

(در اصل، هسیتوگرام ها مقادیری را نشان میدهد که هر ستون دارد و اینکه هر مقدار چقدر تکرار شده است. این کار به ما کمک میکند که یک دید کلی نسبت به ارتباط ویژگی ها با داده های مورد نظر کسب کنیم و در نتیجه تحلیل آسان تر شود.)

```
import matplotlib.pyplot as plt


def draw_hist():
    for columnName in data.columns:
        plt.figure()
        data[columnName].hist()
        plt.xlabel(columnName)
        plt.ylabel('Frequency')
        plt.title('Histogram of {}'.format(columnName))
        plt.show()
draw_hist()
```

در هیستوگرام اول، تشخیص مورد توجه است که مقادیری دارد و میبینیم که بیترین مقدار تکرار شده

در این بخش داده از دست رفته ای نداریم و با استفاده از ماتریس همبستگی داده مورد نظر را پیدا میکنیم و سپس به نرمال کردن داده ها می پردازیم. کد:

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

data.corr()
```

در این بخش با استفاده از کتابخانه seaborn نقشه حرارتی داده ها را ترسیم میکنیم. از طرفی sns.heatmap با در نظر گرفتن ماتریس همبستگی به عنوان ورودی، نقشه حرارتی ایجاد می کند. کد:

```
import seaborn as sns


plt.figure()

f,ax = plt.subplots(figsize=(14, 14))

sns.heatmap(data.corr(), annot=True, cmap='coolwarm', linewidths=.5, fmt=
'.1f',ax=ax)

plt.title("Correlation Matrix")

plt.show()
```

در این بخش تابعی را برای انتخاب متغیرهای مستقل بر اساس همبستگی با سایر متغیرها تعریف می کنیم

این تابع از متغیرهای مستقل "x" و یک لیست از ستون های همبستگی را به عنوان ورودی می گیرد

و همچنین تابع fs_corr برای بررسی ویژگی داده ها وجود دارد. کد:

```
import numpy as np


def by_correlation(x,drop_list_cor):
    """
    selects the features by correlation


    args:
        x (pd.DataFrame):a dataframe of the independent variables
        drop_list_cor (list): a list of the columns believed to have high correlation


    returns:
        a dataframe demonstrating correlation among
    """
    x_1 = x.drop(drop_list_cor,axis = 1 )       # do not modify x, we will use it later
    x_1.head()
    selected_feature_corr=x_1.columns
    fs_corr = np.ones(len(x_1.columns)).astype(int)
    fs_corr = pd.DataFrame(fs_corr, columns = ["Corr"], index=x_1.columns)
```

```
    f,ax = plt.subplots(figsize=(14, 14))
    sns.heatmap(x_1.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
    return x_1,fs_corr


x = data.drop(['diagnosis'], axis=1)
drop_list_cor = ['perimeter_mean','radius_mean','compactness_mean','concave
points_mean','radius_se','perimeter_se','radius_worst','perimeter_worst','compa
ctness_worst','concave points_worst','compactness_se','concave
points_se','texture_worst','area_worst']
x_1,fs_corr = by_correlation(x,drop_list_cor)
                                                    fs_corr = fs_corr.reset_index();fs_corr
```

در این بخش تابع دو پارامتر x و y دارد که به ترتیب نشان دهنده متغیرهای مستقل و متغیر وابسته هستند.

این داده ها را به مجموعه های train و test تقسیم می کند. مجموعه train ۷۰ درصد از داده ها و ۳۰ درصد باقی مانده برای test استفاده می شود و RandomForestClassifier یک نمونه کلاس از پارامترهای پیش‌فرض ایجاد می‌کند و آن را با داده‌های train متناسب می‌کند. و سپس دقت مدل را بررسی می کنیم.

کد:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.model_selection import train_test_split

def test_rf(x,y):
    """
    find correlation tests by random forest

    args:
        x (pd.DataFrame): a dataframe of the independent variables
        y (pd.DataFrame): a dataframe of the dependent variable

    returns:
        a heatmap showing rf results
```

```
    """
    # split data train 70 % and test 30 %
    x_train, x_test, y_train, y_test = train_test_split(x_1, y, test_size=0.3,
random_state=42)

    #random forest classifier with n_estimators=10 (default)
    clf_rf = RandomForestClassifier(random_state=43)
    clr_rf = clf_rf.fit(x_train,y_train)

    ac = accuracy_score(y_test,clf_rf.predict(x_test))
    print('Accuracy is: ',ac)
    cm = confusion_matrix(y_test,clf_rf.predict(x_test))
    sns.heatmap(cm,annot=True,fmt="d")
    return x_train, x_test, y_train, y_test,clr_rf

x_train, x_test, y_train, y_test,clr_rf = test_rf(x_1,y
```

این کد یک تابع train_mlp را تعریف می کند که یک پرسپترون چند لایه (MLP) را روی داده های train می دهد.

۳ لایه داریم: لایه ورودی با ۶۴ نورون - تابع فعال سازی ReLU - یک لایه خروجی با ۱ نورون و تابع فعال سازی سیگموئید

روش کامپایل برای تعیین معیارهای بهینه ساز، تابع ضرر و ارزیابی برای مدل استفاده می شود.

در این مورد، بهینه ساز Adam با نرخ یادگیری مشخص شده، آنتروپی متقاطع باینری به عنوان تابع ضرر و دقت به عنوان متریک ارزیابی استفاده می شود. کد:

```
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion_matrix, f1_score, accuracy_score,
precision_score


# Standardize the features

scaler = StandardScaler()

X_train = scaler.fit_transform(x_train)

X_test = scaler.transform(x_test)


def train_mlp(X_train, y_train, learning_rate, epochs):

   model = tf.keras.Sequential([

      tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),

      tf.keras.layers.Dense(64, activation='relu'),

      tf.keras.layers.Dense(1, activation='sigmoid')
```

```
    ])


    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=epochs, batch_size=32, verbose=1)
return model
```

در این کد یک مدل را به عنوان ورودی می گیرد و عملکرد مدل را در مجموعه test ارزیابی می کند. این تابع معیارهای مختلف عملکرد مانند ضرر تست، دقت تست، ماتریس سردرگمی، امتیاز F1، دقت و دقت را محاسبه می‌کند. کد:

```
def print_result(model):
    print("=" * 50)
    loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
    print(f'Test loss: {loss:.4f}')
    print(f'Test accuracy: {accuracy:.4f}')

    y_pred_prob = model.predict(X_test)
    y_pred = (y_pred_prob > 0.5).astype(int)

    conf_matrix = confusion_matrix(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)

    print("Confusion Matrix:")
    print(conf_matrix)
    print("F1-Score:", f1)
```

```
    print("Accuracy:", accuracy)
    print("Precision:", precision)


for model in models:
(print_result(model
```