In [289…

```
import pandas as pd
import matplotlib.pyplot as plt
```

In [290…

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

In [291…

```
df.head()
```

Out[291]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compact
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 33 columns

In [292…

```
df
```

Out[292]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	

569 rows × 33 columns

In [295…

```
df.head()
```

Out[295]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280

5 rows × 32 columns

Loading [MathJax]/extensions/Safe.js

In [412…

```
df
```

Out[412]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
0	1	17.99	10.38	122.80	1001.0	0.11840	0.277
1	1	20.57	17.77	132.90	1326.0	0.08474	0.078
2	1	19.69	21.25	130.00	1203.0	0.10960	0.159
3	1	11.42	20.38	77.58	386.1	0.14250	0.283
4	1	20.29	14.34	135.10	1297.0	0.10030	0.132
...	...	...	...	...	...	...	
564	1	21.56	22.39	142.00	1479.0	0.11100	0.115
565	1	20.13	28.25	131.20	1261.0	0.09780	0.103
566	1	16.60	28.08	108.30	858.1	0.08455	0.102
567	1	20.60	29.33	140.10	1265.0	0.11780	0.277
568	0	7.76	24.54	47.92	181.0	0.05263	0.043

569 rows × 32 columns

In [296…

```
df.columns
```

Out[296]:

```
Index(['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', 'Unnamed: 32'],
      dtype='object')
```

In [298…

```
mean_features = list(df.columns[1:11])
se_features = list(df.columns[11:21])
worst_features = list(df.columns[21:31])
```

In [299…

```
mean_features.append('diagnosis')
se_features.append('diagnosis')
worst_features.append('diagnosis')
```

In [300…

```
corr = df[mean_features].corr()
corr
```

Out[300]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compact
radius_mean	1.000000	0.323782	0.997855	0.987357	0.170581	
texture_mean	0.323782	1.000000	0.329533	0.321086	-0.023389	
perimeter_mean	0.997855	0.329533	1.000000	0.986507	0.207278	
area_mean	0.987357	0.321086	0.986507	1.000000	0.177028	
smoothness_mean	0.170581	-0.023389	0.207278	0.177028	1.000000	
compactness_mean	0.506124	0.236702	0.556936	0.498502	0.659123	
concavity_mean	0.676764	0.302418	0.716136	0.685983	0.521984	
concave points_mean	0.822529	0.293464	0.850977	0.823269	0.553695	
symmetry_mean	0.147741	0.071401	0.183027	0.151293	0.557775	
fractal_dimension_mean	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	
diagnosis	0.730029	0.415185	0.742636	0.708984	0.358560	

In [301…

```
corr = df[se_features].corr()
corr
```

Out[301]:

	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concav
radius_se	1.000000	0.213247	0.972794	0.951830	0.164514	0.356065	0.3
texture_se	0.213247	1.000000	0.223171	0.111567	0.397243	0.231700	0.1
perimeter_se	0.972794	0.223171	1.000000	0.937655	0.151075	0.416322	0.3
area_se	0.951830	0.111567	0.937655	1.000000	0.075150	0.284840	0.2
smoothness_se	0.164514	0.397243	0.151075	0.075150	1.000000	0.336696	0.2
compactness_se	0.356065	0.231700	0.416322	0.284840	0.336696	1.000000	0.8
concavity_se	0.332358	0.194998	0.362482	0.270895	0.268685	0.801268	1.0
concave points_se	0.513346	0.230283	0.556264	0.415730	0.328429	0.744083	0.7
symmetry_se	0.240567	0.411621	0.266487	0.134109	0.413506	0.394713	0.3
fractal_dimension_se	0.227754	0.279723	0.244143	0.127071	0.427374	0.803269	0.7
diagnosis	0.567134	-0.008303	0.556141	0.548236	-0.067016	0.292999	0.2

In [302…

```
corr = df[worst_features].corr()
corr
```

Loading [MathJax]/extensions/Safe.js

Out[302]:

	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compac
radius_worst	1.000000	0.359921	0.993708	0.984015	0.216574	
texture_worst	0.359921	1.000000	0.365098	0.345842	0.225429	
perimeter_worst	0.993708	0.365098	1.000000	0.977578	0.236775	
area_worst	0.984015	0.345842	0.977578	1.000000	0.209145	
smoothness_worst	0.216574	0.225429	0.236775	0.209145	1.000000	
compactness_worst	0.475820	0.360832	0.529408	0.438296	0.568187	
concavity_worst	0.573975	0.368366	0.618344	0.543331	0.518523	
concave points_worst	0.787424	0.359755	0.816322	0.747419	0.547691	
symmetry_worst	0.243529	0.233027	0.269493	0.209146	0.493838	
fractal_dimension_worst	0.093492	0.219122	0.138957	0.079647	0.617624	
diagnosis	0.776454	0.456903	0.782914	0.733825	0.421465	

In [ ]:

```

```

In [303…

```python
prediction_var = ['perimeter_mean', 'compactness_mean', 'concavity_mean']
```

In [304…

```python
from sklearn.model_selection import train_test_split
```

In [336…

```python
train, test = train_test_split(df, test_size = 0.15, random_state= 1)
```

In [339…

```python
train_x = train[prediction_var]
train_y = train['diagnosis']

test_x = test[prediction_var]
test_y = test['diagnosis']
```

In [307…

```python
 #Import multiple options, to enable us to try out different classifiers

from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
```

In [342…

```python
model = MLPClassifier()

clf = MLPClassifier(hidden_layer_sizes=(6,5),
                    random_state=5,
                    verbose=True,
                    learning_rate_init=0.01)
```

In [343…

```python
clf.fit(train_x,train_y)
```

Loading [MathJax]/extensions/Safe.js

```
Iteration 1, loss = 22.76048903
Iteration 2, loss = 22.68155216
Iteration 3, loss = 21.39738874
Iteration 4, loss = 16.03174765
Iteration 5, loss = 9.39400606
Iteration 6, loss = 4.06393652
Iteration 7, loss = 1.47993455
Iteration 8, loss = 0.77384071
Iteration 9, loss = 0.76774639
Iteration 10, loss = 0.76252621
Iteration 11, loss = 0.75819762
Iteration 12, loss = 0.75391217
Iteration 13, loss = 0.75002932
Iteration 14, loss = 0.74632568
Iteration 15, loss = 0.74297240
Iteration 16, loss = 0.73964392
Iteration 17, loss = 0.73662063
Iteration 18, loss = 0.73333277
Iteration 19, loss = 0.73047745
Iteration 20, loss = 0.72751711
Iteration 21, loss = 0.72481400
Iteration 22, loss = 0.72203330
Iteration 23, loss = 0.71926583
Iteration 24, loss = 0.71660964
Iteration 25, loss = 0.71407071
Iteration 26, loss = 0.71155441
Iteration 27, loss = 0.70910678
Iteration 28, loss = 0.70676498
Iteration 29, loss = 0.70449899
Iteration 30, loss = 0.70231651
Iteration 31, loss = 0.70021287
Iteration 32, loss = 0.69820762
Iteration 33, loss = 0.69623828
Iteration 34, loss = 0.69436729
Iteration 35, loss = 0.69260459
Iteration 36, loss = 0.69075091
Iteration 37, loss = 0.68922598
Iteration 38, loss = 0.68770472
Iteration 39, loss = 0.68618519
Iteration 40, loss = 0.68467964
Iteration 41, loss = 0.68342743
Iteration 42, loss = 0.68211071
Iteration 43, loss = 0.68080541
Iteration 44, loss = 0.67961758
Iteration 45, loss = 0.67844581
Iteration 46, loss = 0.67749514
Iteration 47, loss = 0.67637674
Iteration 48, loss = 0.67546891
Iteration 49, loss = 0.67460526
Iteration 50, loss = 0.67365477
Iteration 51, loss = 0.67271830
Iteration 52, loss = 0.67194987
Iteration 53, loss = 0.67107800
Iteration 54, loss = 0.67032834
Iteration 55, loss = 0.66957063
Iteration 56, loss = 0.66883527
Iteration 57, loss = 0.66820962
Iteration 58, loss = 0.66749717
Iteration 59, loss = 0.66694597
Iteration 60, loss = 0.66647709
Iteration 61, loss = 0.66589984
Iteration 62, loss = 0.66550075
Iteration 63, loss = 0.66494311
Iteration 64, loss = 0.66457847
Iteration 65, loss = 0.66416500
Iteration 66, loss = 0.66375391
Iteration 67, loss = 0.66334230
Iteration 68, loss = 0.66306404
Iteration 69, loss = 0.66268567
Iteration 70, loss = 0.66239802
Iteration 71, loss = 0.66204637
Iteration 72, loss = 0.66185327
Iteration 73, loss = 0.66148404
Iteration 74, loss = 0.66124674
Iteration 75, loss = 0.66102454
Iteration 76, loss = 0.66081638
Iteration 77, loss = 0.66062474
Iteration 78, loss = 0.66042610
Iteration 79, loss = 0.66028069
Iteration 80, loss = 0.66012292
Iteration 81, loss = 0.66000451
Iteration 82, loss = 0.65987291
Iteration 83, loss = 0.65977724
Iteration 84, loss = 0.65967691
Iteration 85, loss = 0.65953616
Iteration 86, loss = 0.65945482
Iteration 87, loss = 0.65937944
Iteration 88, loss = 0.65931332
Iteration 89, loss = 0.65924677
Iteration 90, loss = 0.65916795
Iteration 91, loss = 0.65910634
Iteration 92, loss = 0.65907757
Iteration 93, loss = 0.65899105
Iteration 94, loss = 0.65896684
Iteration 95, loss = 0.65888264
Iteration 96, loss = 0.65882415
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

Out[343]:

```
▼                         MLPClassifier
MLPClassifier(hidden_layer_sizes=(6, 5), learning_rate_init=0.01,
              random_state=5, verbose=True)
```

In [405…

```
test_1 = test_x.iloc[1,0:3]
```

In [406…

```
test_1
```

Out[406]:

```
perimeter_mean      85.9800
compactness_mean     0.1231
concavity_mean       0.1226
Name: 47, dtype: float64
```

In [411…

```
df.loc[66,'diagnosis']
```

Out[411]:

```
0
```

In [404…

```
clf.predict([test_1])
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClassifier was fitted with feature names
  warnings.warn(
```

Out[404]:

```
array([0], dtype=int64)
```

In [365…

```
test_2 = test_x.iloc[1,0:3]
test_3 = test_x.iloc[45,0:3]
```

Loading [MathJax]/extensions/Safe.js

In [366…

```
test_2
```

Out[366]:

```
perimeter_mean      85.9800
compactness_mean     0.1231
concavity_mean       0.1226
Name: 47, dtype: float64
```

In [367…

```
test_3
```

Out[367]:

```
perimeter_mean      103.2000
compactness_mean      0.2284
concavity_mean        0.2448
Name: 257, dtype: float64
```

In [414…

```
df.loc[567,'diagnosis']
```

Out[414]:

```
1
```

In [421…

```
df.loc[566,'diagnosis']
```

Out[421]:

```
1
```

In [422…

```
clf.predict([test_2])
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClassifier was fitted with feature names
  warnings.warn(
```

Out[422]:

```
array([0], dtype=int64)
```

In [423…

```
clf.predict([test_3])
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClassifier was fitted with feature names
  warnings.warn(
```

Out[423]:

```
array([0], dtype=int64)
```

In [424…

```
from sklearn import metrics


y_true = test_y
y_pred = clf.predict(test_x)
```

In [425…

```
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score

recall = recall_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)

print("The recall score is ", "%.2f" %recall)
print("The precision score is ", "%.2f" %precision)
```

```
The recall score is  0.00
The precision score is  0.00
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [426…

```
F1 = 2 * (precision * recall) / (precision + recall)
```

Loading [MathJax]/extensions/Safe.js

```
%F1)
```

```
nan
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_14680\242740335.py:1: RuntimeWarning: invalid v
alue encountered in double_scalars
  F1 = 2 * (precision * recall) / (precision + recall)
```

In [427…

```
from sklearn.metrics import f1_score

f1_score = f1_score(y_true, y_pred)

print("The f1 score is ", "%.2f" %f1_score)
```

```
The f1 score is  0.00
```

In [428…

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score

accuracy = accuracy_score(y_true, y_pred)
AUC = roc_auc_score(y_true, y_pred)

print("The accuracy is ", "%.2f" %accuracy)
print("The AUC is ", "%.2f" %AUC)
```

```
The accuracy is  0.60
The AUC is  0.50
```

In [429…

```
from sklearn.metrics import confusion_matrix



confusion_matrix = confusion_matrix(y_true, y_pred)
```

In [430…

```
confusion_matrix
```

Out[430]:

```
array([[52,  0],
       [34,  0]], dtype=int64)
```

In [ ]:

Loading [MathJax]/extensions/Safe.js