!pip install seaborn==0.11.0 → Collecting seaborn==0.11.0 Downloading seaborn-0.11.0-py3-none-any.whl (283 kB) - 283.1/283.1 kB 3.6 MB/s eta 0:00:00 Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.10/dist-packages (from seaborn==0.11.0) (1.25.2) Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.10/dist-packages (from seaborn==0.11.0) (1.11.4) Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.10/dist-packages (from seaborn==0.11.0) (2.0.3) Requirement already satisfied: matplotlib>=2.2 in /usr/local/lib/python3.10/dist-packages (from seaborn==0.11.0) (3.7.1) Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (1.2.1) Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (0.12.1) Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (4.53.1) Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (1.4.5) Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (24.1) Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (9.4.0) Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (3.1.2) Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2->seaborn==0.11.0) (2.8.2) Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.23->seaborn==0.11.0) (2023.4) Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.23->seaborn==0.11.0) (2024.1) Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn==0.11.0) (1.16.0) Installing collected packages: seaborn Attempting uninstall: seaborn Found existing installation: seaborn 0.13.1 Uninstalling seaborn-0.13.1: Successfully uninstalled seaborn-0.13.1 Successfully installed seaborn-0.11.0 import pandas as pd import pandas.io.json as pd json

```
import ison
import pandas as pd # Import pandas at the top level
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier , GradientBoostingClassifier
from sklearn.naive bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion matrix
from sklearn.metrics import classification report
from sklearn.model_selection import train_test_split
from \ sklearn.ensemble \ import \ Random Forest Classifier \ , \ Gradient Boosting Classifier
# Use pd.json_normalize directly
import pandas as pd
local = '/content/data.csv'
df = pd.read_csv(local, encoding='latin-1', index_col=0)
```

```
columns list = [
 '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',
 'diagnosis']
df = df[columns list]
cm = sns.light_palette("green", as_cmap=True)
df.head(30).style.background_gradient(cmap=cm)
```



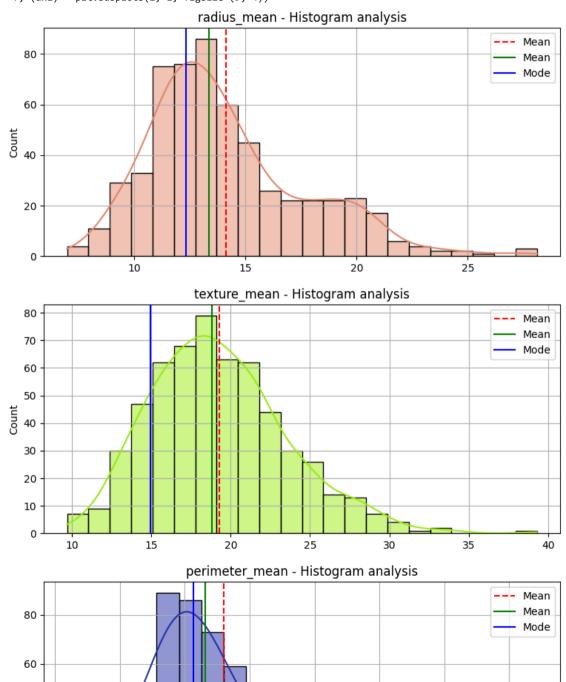
	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
id								
842302	17.990000	10.380000	122.800000	1001.000000	0.118400	0.277600	0.300100	0.147100
842517	20.570000	17.770000	132.900000	1326.000000	0.084740	0.078640	0.086900	0.070170
84300903	19.690000	21.250000	130.000000	1203.000000	0.109600	0.159900	0.197400	0.127900
84348301	11.420000	20.380000	77.580000	386.100000	0.142500	0.283900	0.241400	0.105200
84358402	20.290000	14.340000	135.100000	1297.000000	0.100300	0.132800	0.198000	0.104300
843786	12.450000	15.700000	82.570000	477.100000	0.127800	0.170000	0.157800	0.080890
844359	18.250000	19.980000	119.600000	1040.000000	0.094630	0.109000	0.112700	0.074000
84458202	13.710000	20.830000	90.200000	577.900000	0.118900	0.164500	0.093660	0.059850
844981	13.000000	21.820000	87.500000	519.800000	0.127300	0.193200	0.185900	0.093530
84501001	12.460000	24.040000	83.970000	475.900000	0.118600	0.239600	0.227300	0.085430
845636	16.020000	23.240000	102.700000	797.800000	0.082060	0.066690	0.032990	0.033230
84610002	15.780000	17.890000	103.600000	781.000000	0.097100	0.129200	0.099540	0.066060
846226	19.170000	24.800000	132.400000	1123.000000	0.097400	0.245800	0.206500	0.111800
846381	15.850000	23.950000	103.700000	782.700000	0.084010	0.100200	0.099380	0.053640
84667401	13.730000	22.610000	93.600000	578.300000	0.113100	0.229300	0.212800	0.080250
84799002	14.540000	27.540000	96.730000	658.800000	0.113900	0.159500	0.163900	0.073640
848406	14.680000	20.130000	94.740000	684.500000	0.098670	0.072000	0.073950	0.052590
84862001	16.130000	20.680000	108.100000	798.800000	0.117000	0.202200	0.172200	0.102800
849014	19.810000	22.150000	130.000000	1260.000000	0.098310	0.102700	0.147900	0.094980
8510426	13.540000	14.360000	87.460000	566.300000	0.097790	0.081290	0.066640	0.047810
8510653	13.080000	15.710000	85.630000	520.000000	0.107500	0.127000	0.045680	0.031100
8510824	9.504000	12.440000	60.340000	273.900000	0.102400	0.064920	0.029560	0.020760
8511133	15.340000	14.260000	102.500000	704.400000	0.107300	0.213500	0.207700	0.097560
851509	21.160000	23.040000	137.200000	1404.000000	0.094280	0.102200	0.109700	0.086320
852552	16.650000	21.380000	110.000000	904.600000	0.112100	0.145700	0.152500	0.091700
852631	17.140000	16.400000	116.000000	912.700000	0.118600	0.227600	0.222900	0.140100
852763	14.580000	21.530000	97.410000	644.800000	0.105400	0.186800	0.142500	0.087830
852781	18.610000	20.250000	122.100000	1094.000000	0.094400	0.106600	0.149000	0.077310

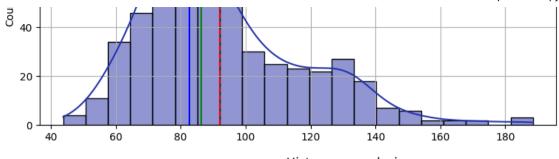
```
Cancer prediction.ipynb - Colab
                    15.300000
                                  25.270000
                                                               732.400000
                                                                                    0.108200
                                                                                                       0.169700
                                                                                                                        0.168300
                                                                                                                                     0.087510
       852973
                                                  102.400000
       853201
                   17.570000
                                  15.050000
                                                  115.000000
                                                               955.100000
                                                                                    0.098470
                                                                                                       0.115700
                                                                                                                        0.098750
                                                                                                                                     0.079530
df.shape
```

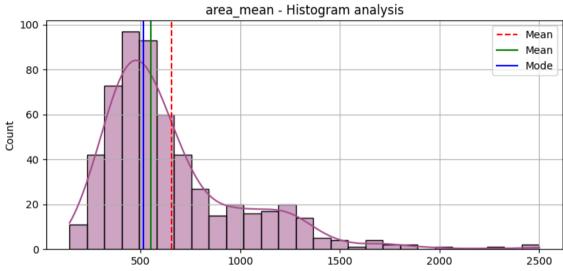
```
pd.isnull(df).sum() > 0
    radius mean
                                False
     texture mean
                                False
     perimeter mean
                                False
     area mean
                                False
     smoothness_mean
                                False
                                False
     compactness_mean
                                False
     concavity_mean
     concave points_mean
                                False
                                False
     symmetry mean
     fractal_dimension_mean
                                False
     radius se
                                False
                                False
     texture_se
                                False
     perimeter_se
     area_se
                                False
     smoothness se
                                False
                                False
     compactness se
                                False
     concavity se
     concave points_se
                                False
                                False
     symmetry_se
                                False
     fractal_dimension_se
                                False
     radius_worst
     texture_worst
                                False
     perimeter worst
                                False
     area_worst
                                False
     smoothness worst
                                False
     compactness_worst
                                False
                                False
     concavity_worst
     concave points_worst
                                False
     symmetry worst
                                False
     fractal_dimension_worst
                                False
     diagnosis
                                False
     dtype: bool
features = list(df.columns)
features = features[0:-1]
features
    ['radius_mean',
      'texture_mean',
      'perimeter_mean',
      'area_mean',
      'smoothness mean',
      'compactness_mean',
```

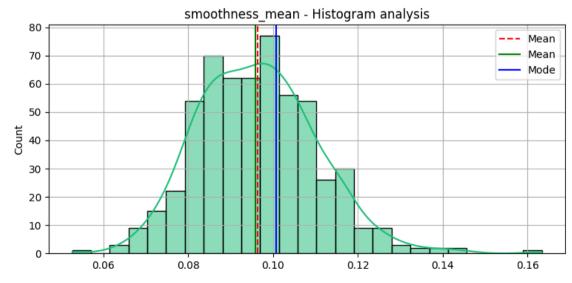
```
'symmetry mean'.
       'fractal_dimension_mean',
       'radius se',
       'texture_se',
       'perimeter_se',
       'area_se',
       'smoothness se',
       'compactness se',
       'concavity_se',
       'concave points_se',
       'symmetry se',
       'fractal_dimension_se',
       'radius_worst',
       'texture_worst',
       'perimeter worst',
       'area_worst',
       'smoothness worst',
       'compactness worst',
       'concavity_worst',
       'concave points_worst',
       'symmetry_worst',
       'fractal_dimension_worst']
import random
def get random color():
   r1 = lambda: random.randint(0,255)
   return '#%02X%02X%02X' % (r1(),r1(),r1())
def get histplot central tendency(df: dict, fields: list):
   for field in fields:
       f, (ax1) = plt.subplots(1, 1, figsize=(9, 4))
       v_dist_1 = df[field].values
       sns.histplot(v_dist_1, ax=ax1, color=get_random_color(), kde=True)
       mean=df[field].mean()
       median=df[field].median()
       mode=df[field].mode().values[0]
       ax1.axvline(mean, color='r', linestyle='--', label="Mean")
       ax1.axvline(median, color='g', linestyle='-', label="Mean")
       ax1.axvline(mode, color='b', linestyle='-', label="Mode")
       ax1.legend()
       plt.grid()
       plt.title(f"{field} - Histogram analysis")
get histplot central tendency(df, features)
```

<ipython-input-17-b8084809e78d>:10: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot ir
f, (ax1) = plt.subplots(1, 1, figsize=(9, 4))

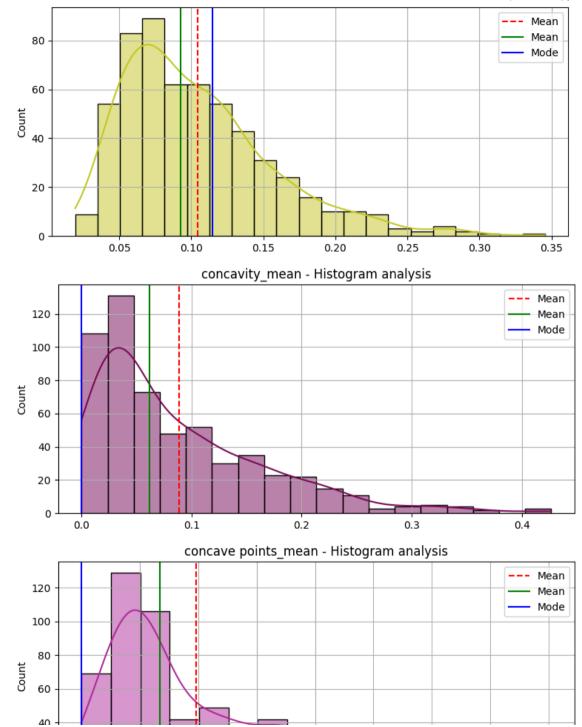


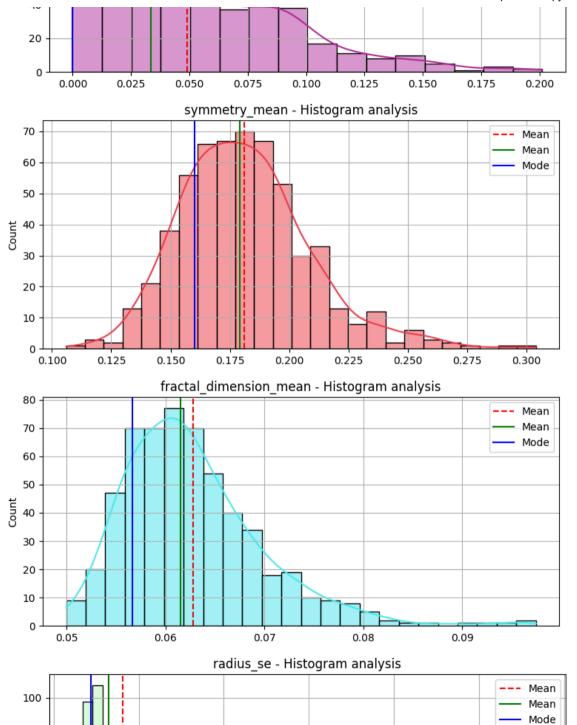


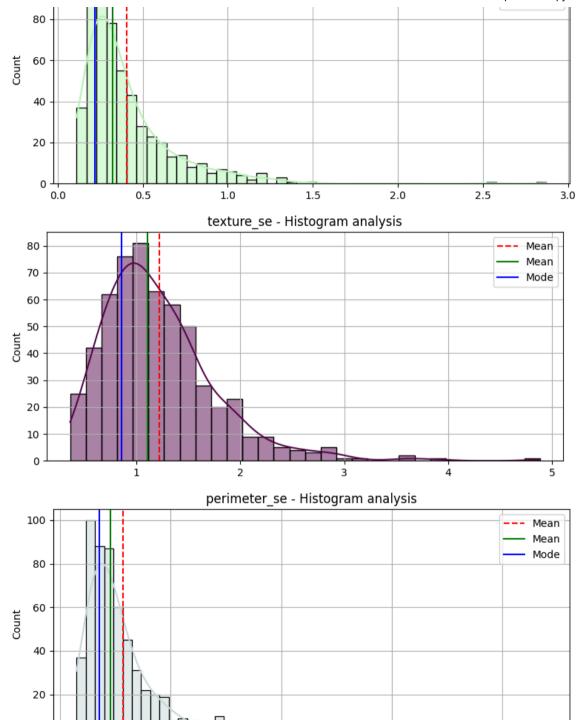


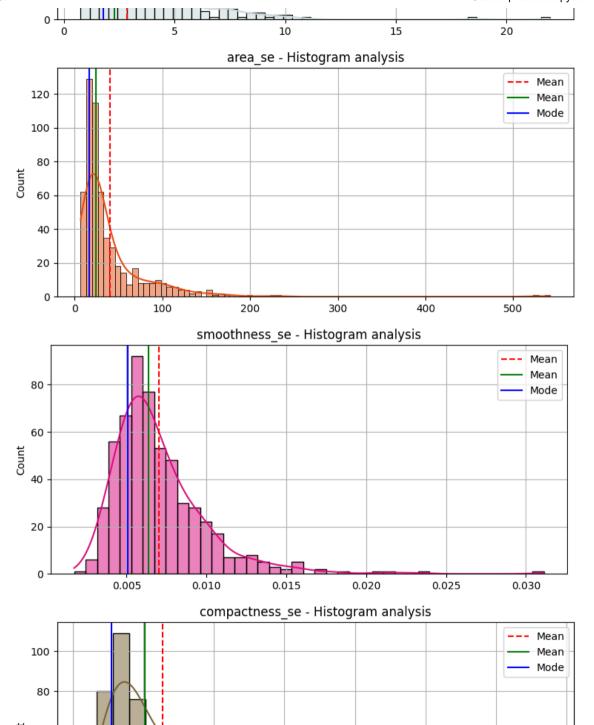


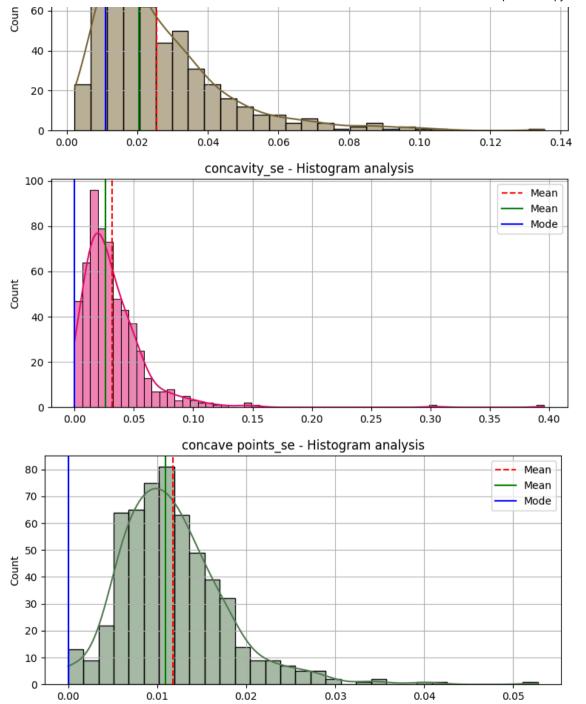
compactness\_mean - Histogram analysis



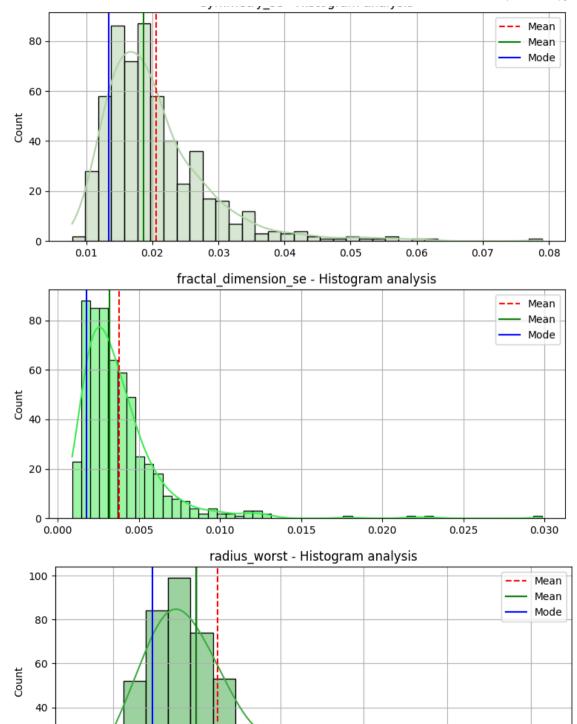


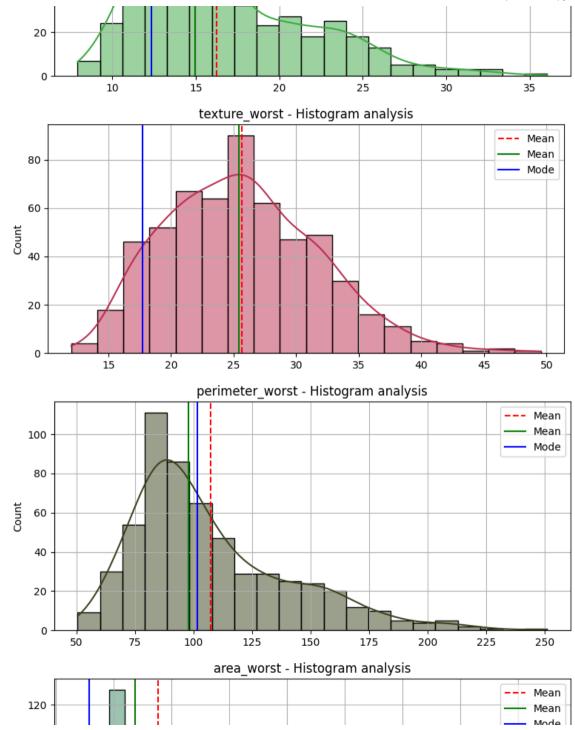


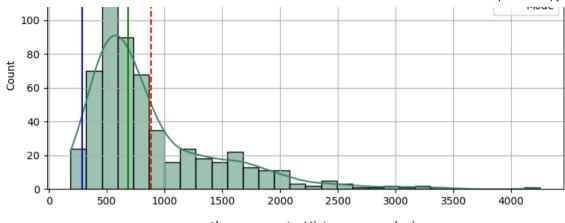


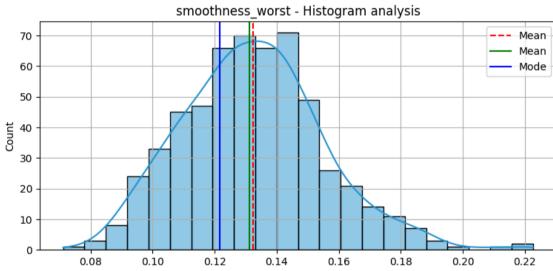


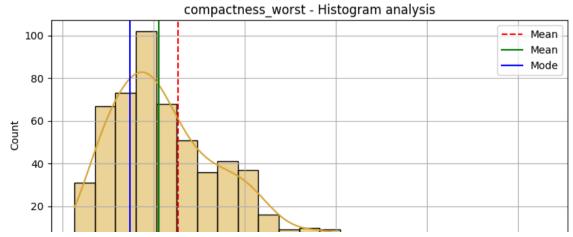
symmetry se - Histogram analysis

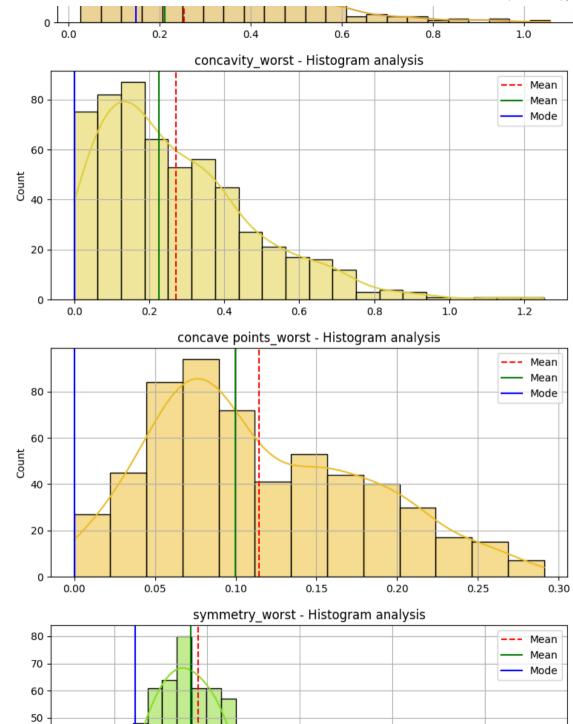


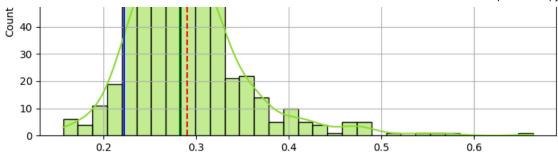


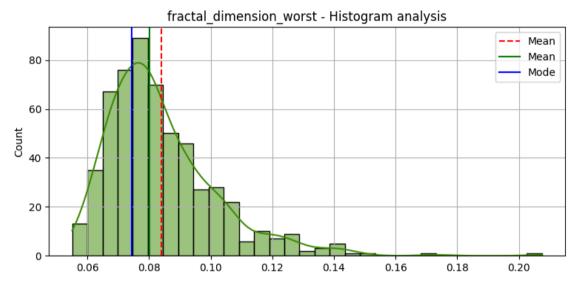












pd.isnull(df).sum() > 0

<b>→</b>	radius_mean texture_mean perimeter_mean	False False False
	area_mean	False
	smoothness_mean	False
	compactness_mean	False
	concavity_mean	False
	concave points_mean	False
	symmetry_mean	False
	fractal_dimension_mean	False
	radius_se	False
	texture_se	False
	perimeter_se	False
	area_se	False
	smoothness_se	False
	compactness_se	False
	concavity_se	False
	concave points_se	False

symmetry_se	False
<pre>fractal_dimension_se</pre>	False
radius_worst	False
texture_worst	False
perimeter_worst	False
area_worst	False
smoothness_worst	False
compactness_worst	False
concavity_worst	False
concave points_worst	False
symmetry_worst	False
<pre>fractal_dimension_worst</pre>	False
diagnosis	False
dtype: bool	

df['diagnosis'] = df.diagnosis.astype("category").cat.codes
df.head(10).style.background\_gradient(cmap=cm)



	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	<pre>concave points_mean</pre>
id								
842302	17.990000	10.380000	122.800000	1001.000000	0.118400	0.277600	0.300100	0.147100
842517	20.570000	17.770000	132.900000	1326.000000	0.084740	0.078640	0.086900	0.070170
84300903	19.690000	21.250000	130.000000	1203.000000	0.109600	0.159900	0.197400	0.127900
84348301	11.420000	20.380000	77.580000	386.100000	0.142500	0.283900	0.241400	0.105200
84358402	20.290000	14.340000	135.100000	1297.000000	0.100300	0.132800	0.198000	0.104300
843786	12.450000	15.700000	82.570000	477.100000	0.127800	0.170000	0.157800	0.080890
844359	18.250000	19.980000	119.600000	1040.000000	0.094630	0.109000	0.112700	0.074000
84458202	13.710000	20.830000	90.200000	577.900000	0.118900	0.164500	0.093660	0.059850
844981	13.000000	21.820000	87.500000	519.800000	0.127300	0.193200	0.185900	0.093530
84501001	12.460000	24.040000	83.970000	475.900000	0.118600	0.239600	0.227300	0.085430
4								<b>&gt;</b>

```
X = df[features]
y = df['diagnosis']
print('Length X:', X.shape)
print('Length Y:', y.shape)
```

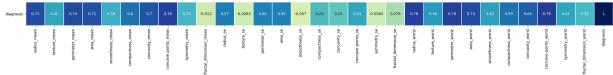
Length X: (569, 30) Length Y: (569,)

```
def get_headmap(df: dict):
    corr = df.corr()
    plt.figure(figsize=(35, 35))
    sns.heatmap(corr, annot=True, cmap="YlGnBu", linewidths=0.1, annot_kws={"fontsize":10})
    plt.title("Correlation house prices - return rate")

get_headmap(df)
```



	Correlation house prices - return rate																														
radius_mean -	1	0.32	1	0.99	0.17	0.51	0.68	0.82	0.15	-0.31	0.68	-0.097	0.67	0.74	-0.22	0.21	0.19	0.38	-0.1	-0.043	0.97	0.3	0.97	0.94	0.12	0.41	0.53	0.74	0.16	0.0071	0.73
texture_mean -	0.32	1	0.33	0.32	-0.023	0.24	0.3	0.29	0.071	-0.076	0.28	0.39	0.28	0.26	0.0066	0.19	0.14	0.16	0.0091	0.054	0.35	0.91	0.36	0.34	0.078	0.28	0.3	0.3	0.11	0.12	0.42
perimeter_mean -	1	0.33	1	0.99	0.21	0.56	0.72	0.85	0.18	-0.26	0.69	-0.087	0.69	0.74	-0.2	0.25	0.23	0.41	-0.082	-0.0055	0.97	0.3	0.97	0.94	0.15	0.46	0.56		0.19	0.051	0.74
area_mean -	0.99	0.32	0.99	1	0.18	0.5	0.69	0.82	0.15	-0.28	0.73	-0.066	0.73	0.8	-0.17	0.21	0.21	0.37	-0.072	-0.02	0.96	0.29	0.96	0.96	0.12	0.39	0.51	0.72	0.14	0.0037	0.71
smoothness_mean -	0.17	-0.023	0.21	0.18	1	0.66	0.52	0.55		0.58	0.3	0.068	0.3	0.25	0.33	0.32	0.25	0.38	0.2	0.28	0.21	0.036	0.24	0.21	0.81	0.47	0.43	0.5		0.5	0.36
compactness_mean -	0.51	0.24	0.56	0.5	0.66	1	0.88	0.83	0.6	0.57	0.5	0.046	0.55	0.46	0.14	0.74	0.57	0.64	0.23	0.51	0.54	0.25	0.59	0.51	0.57	0.87	0.82	0.82	0.51	0.69	0.6
concavity_mean -	0.68	0.3	0.72	0.69	0.52	0.88	1	0.92	0.5	0.34	0.63	0.076	0.66	0.62	0.099	0.67	0.69	0.68	0.18	0.45	0.69	0.3	0.73	0.68	0.45	0.75	0.88	0.86	0.41	0.51	0.7
concave points_mean -	0.82	0.29	0.85	0.82		0.83	0.92	1		0.17		0.021	0.71		0.028	0.49	0.44		0.095	0.26		0.29	0.86	0.81		0.67	0.75				0.78
symmetry_mean -	0.15	0.071	0.18	0.15		0.6	0.5	0.46	1		0.3	0.13		0.22	0.19	0.42	0.34		0.45	0.33	0.19	0.091	0.22	0.18		0.47	0.43			0.44	0.33
fractal_dimension_mean -	-0.31	-0.076	-0.26	-0.28	0.58	0.57	0.34	0.17	0.48	1	0.00011	0.16	0.04	-0.09	0.4	0.56	0.45	0.34	0.35	0.69	-0.25	-0.051	-0.21	-0.23	0.5	0.46	0.35	0.18		0.77	-0.013
radius_se -	0.68	0.28	0.69	0.73	0.3	0.5	0.63	0.7	0.3	0.00011	1	0.21	0.97	0.95	0.16	0.36	0.33	0.51	0.24	0.23	0.72	0.19	0.72	0.75	0.14	0.29	0.38	0.53	0.095	0.05	0.57
texture_se -	-0.097	0.39	-0.087	-0.066	0.068	0.046	0.076	0.021	0.13	0.16	0.21	1	0.22	0.11	0.4	0.23	0.19	0.23	0.41	0.28	-0.11	0.41	-0.1	-0.083	-0.074	-0.092	-0.069	-0.12	-0.13	-0.046	-0.0083
perimeter_se -	0.67	0.28	0.69	0.73	0.3	0.55	0.66	0.71	0.31	0.04	0.97	0.22	1	0.94	0.15	0.42	0.36	0.56	0.27	0.24	0.7	0.2	0.72	0.73	0.13	0.34	0.42	0.55	0.11	0.085	0.56
area_se -	0.74	0.26	0.74	0.8	0.25	0.46	0.62	0.69	0.22	-0.09	0.95	0.11	0.94	1	0.075	0.28	0.27	0.42	0.13	0.13	0.76	0.2	0.76	0.81	0.13	0.28	0.39		0.074	0.018	0.55
smoothness_se -	-0.22	0.0066	-0.2	-0.17		0.14	0.099	0.028	0.19	0.4	0.16	0.4	0.15	0.075	1	0.34	0.27	0.33	0.41	0.43	-0.23	-0.075	-0.22	-0.18	0.31	-0.056	-0.058	-0.1	-0.11	0.1	-0.067
compactness_se -	0.21	0.19	0.25	0.21		0.74	0.67	0.49	0.42	0.56	0.36	0.23	0.42	0.28	0.34	1	0.8	0.74	0.39	0.8	0.2	0.14	0.26	0.2	0.23	0.68	0.64	0.48	0.28	0.59	0.29
concavity_se -	0.19	0.14	0.23	0.21	0.25	0.57	0.69	0.44	0.34	0.45		0.19	0.36	0.27	0.27	0.8	1	0.77	0.31	0.73	0.19	0.1	0.23	0.19	0.17	0.48	0.66		0.2	0.44	0.25
concave points_se -	0.38	0.16	0.41	0.37	0.38	0.64	0.68	0.62		0.34	0.51	0.23	0.56	0.42	0.33	0.74	0.77	1	0.31	0.61		0.087	0.39	0.34	0.22	0.45	0.55	0.6	0.14	0.31	0.41
symmetry_se -	-0.1	0.0091	-0.082	-0.072	0.2	0.23	0.18	0.095			0.24	0.41	0.27	0.13	0.41	0.39	0.31	0.31	1	0.37	-0.13	-0.077	-0.1	-0.11	-0.013	0.06	0.037	-0.03	0.39	0.078	-0.0065
fractal_dimension_se -	-0.043	0.054	-0.0055	-0.02	0.28	0.51	0.45	0.26	0.33	0.69	0.23	0.28	0.24	0.13	0.43	0.8	0.73	0.61	0.37	1	-0.037	-0.0032	-0.001	-0.023	0.17	0.39	0.38	0.22	0.11	0.59	0.078
radius_worst -	0.97	0.35	0.97	0.96	0.21	0.54	0.69	0.83	0.19	-0.25	0.72	-0.11	0.7	0.76	-0.23	0.2	0.19	0.36	-0.13	-0.037	1	0.36	0.99	0.98	0.22	0.48	0.57	0.79	0.24	0.093	0.78
texture_worst -	0.3	0.91	0.3	0.29	0.036	0.25	0.3	0.29	0.091	-0.051	0.19	0.41	0.2	0.2	-0.075	0.14	0.1	0.087	-0.077	-0.0032	0.36	1	0.37	0.35	0.23	0.36	0.37	0.36	0.23	0.22	0.46
perimeter_worst -	0.97	0.36	0.97	0.96	0.24	0.59	0.73	0.86	0.22	-0.21	0.72	-0.1	0.72	0.76	-0.22	0.26	0.23	0.39	-0.1	-0.001	0.99	0.37	1	0.98	0.24	0.53	0.62	0.82	0.27	0.14	0.78
area_worst -	0.94	0.34	0.94	0.96	0.21	0.51	0.68	0.81	0.18	-0.23	0.75	-0.083	0.73	0.81	-0.18	0.2	0.19	0.34	-0.11	-0.023	0.98	0.35	0.98	1	0.21	0.44	0.54		0.21	0.08	0.73
smoothness_worst -	0.12	0.078	0.15	0.12	0.81	0.57	0.45	0.45	0.43	0.5	0.14	-0.074	0.13	0.13	0.31	0.23	0.17	0.22	-0.013	0.17	0.22	0.23	0.24	0.21	1	0.57	0.52	0.55	0.49	0.62	0.42
compactness_worst -	0.41	0.28	0.46	0.39	0.47	0.87	0.75	0.67	0.47		0.29	-0.092	0.34	0.28	-0.056	0.68	0.48	0.45	0.06	0.39	0.48	0.36	0.53	0.44	0.57	1	0.89	0.8	0.61	0.81	0.59
concavity_worst -	0.53	0.3	0.56	0.51		0.82	0.88	0.75				-0.069	0.42		-0.058	0.64	0.66	0.55	0.037	0.38		0.37	0.62	0.54	0.52	0.89	1			0.69	0.66
concave points_worst -	0.74	0.3	0.77	0.72		0.82	0.86	0.91		0.18	0.53	-0.12	0.55	0.54	-0.1	0.48	0.44	0.6	-0.03	0.22	0.79	0.36	0.82	0.75	0.55	0.8	0.86	1		0.51	0.79
symmetry_worst -	0.16	0.11	0.19	0.14	0.39	0.51	0.41	0.38	0.7	0.33	0.095	-0.13	0.11	0.074	-0.11	0.28	0.2	0.14	0.39	0.11	0.24	0.23	0.27	0.21	0.49	0.61	0.53	0.5	1	0.54	0.42
fractal_dimension_worst -		0.12	0.051	0.0037	0.5	0.69	0.51	0.37	0.44	0.77	0.05	-0.046	0.085	0.018	0.1	0.59	0.44	0.31	0.078	0.59	0.093	0.22	0.14	0.08	0.62	0.81	0.69		0.54	1	0.32



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
def KNN(neighbors, X, y):
    model_KNN = KNeighborsClassifier(n_neighbors = 3)
    model_KNN.fit(X , y)
    return model_KNN
np.random.seed(1000)
KNN_predict = KNN(4, X_train, y_train)
from xgboost import plot_importance
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from \ sklearn.ensemble \ import \ Random Forest Classifier
from \ sklearn.neighbors \ import \ KNeighbors Classifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```

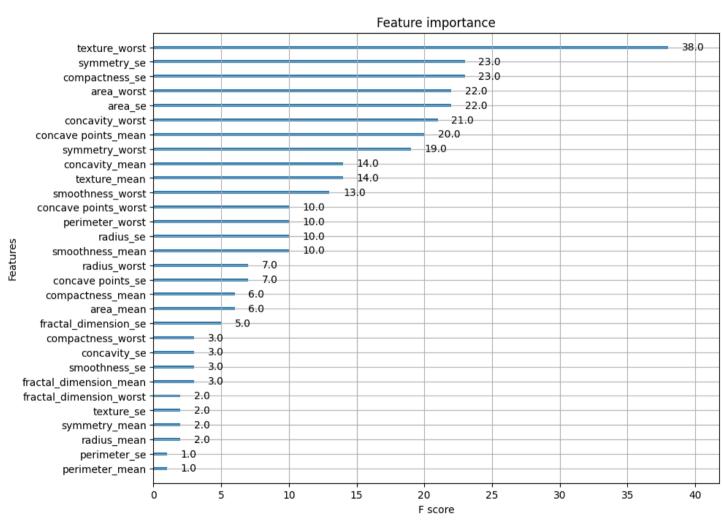
```
from xgboost import XGBClassifier
def machine_learning_algorithms(train_X, train_y):
   estimador = 100
    model_xgb = XGBClassifier(learning_rate=0.3, n_estimators=300, max_depth=14)
    model xgb.fit(train X , train y)
   model SVC = SVC()
    model SVC.fit(train X , train y)
   model GBC = GradientBoostingClassifier()
    model GBC.fit(train X , train y)
    model KNN v2 = KNeighborsClassifier(n neighbors = 2)
    model KNN v2.fit(train X , train y)
    model KNN v4 = KNeighborsClassifier(n neighbors = 4)
    model_KNN_v4.fit(train_X , train_y)
    model_KNN_v6 = KNeighborsClassifier(n_neighbors = 6)
    model KNN v6.fit(train X , train y)
    model_KNN_v8 = KNeighborsClassifier(n_neighbors = 8)
    model KNN v8.fit(train X , train y)
    model LR = LogisticRegression(solver="lbfgs")
   model_LR.fit(train_X , train_y)
    model RFC = RandomForestClassifier(criterion='gini', max depth=None, max features=8, max leaf nodes=None,
                                      n_estimators=100)
    model RFC.fit(train X , train y)
    return [model_xgb, model_SVC, model_GBC, model_KNN_v2, model_KNN_v4, model_KNN_v6, model_KNN_v8,
           model LR, model RFC]
np.random.seed(1000)
algoritms = machine_learning_algorithms(X_train, y_train)
     /usr/local/lib/python3.10/dist-packages/sklearn/linear model/ logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
     STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
     Increase the number of iterations (max_iter) or scale the data as shown in:
          https://scikit-learn.org/stable/modules/preprocessing.html
     Please also refer to the documentation for alternative solver options:
          https://scikit-learn.org/stable/modules/linear model.html#logistic-regression
       n iter i = check optimize result(
score = []
total = []
classifier = ['XGBoost', 'SVG', 'GBC', 'KNN v2', 'KNN v4', 'KNN v6', 'KNN v8', 'LR', 'RandomForest']
for index, name in enumerate(classifier):
   score_train = algoritms[index].score(X_train, y_train) * 100
    score tests = algoritms[index].score(X test. v test) * 100
```

```
total.append([name, score_train, score_tests])

df_result = pd.DataFrame(total, columns = ['Model', 'Train score', 'Test score'])

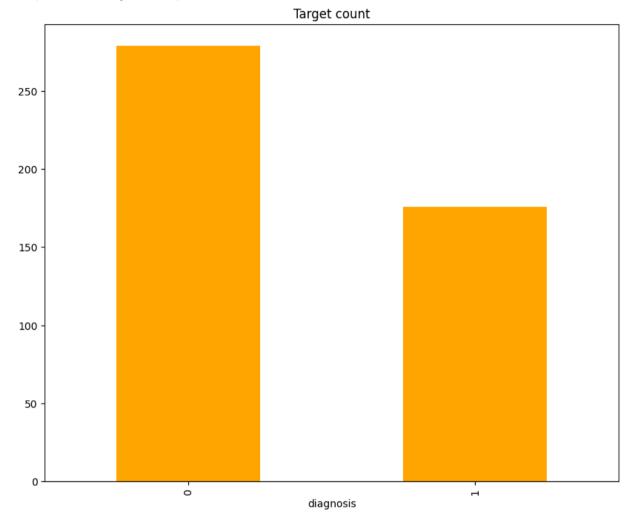
plt.rcParams["figure.figsize"] = (10, 8)
plot_importance(algoritms [0], max_num_features=100)
plt.show()
```





y\_train.value\_counts().plot(kind='bar', color='orange')
plt.title('Target count')

→ Text(0.5, 1.0, 'Target count')



ax = df\_result.plot.bar(x='Model', y='Train score')
ax.set\_ylabel("Score")

→ Text(0, 0.5, 'Score')

