DATA

```
import framework
from sklearn.datasets import load_breast_cancer

df = load_breast_cancer(as_frame=True).frame
df.head()
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030

   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst
area  \
0                 0.07871  ...          17.33           184.60
2019.0
1                 0.05667  ...          23.41           158.80
1956.0
2                 0.05999  ...          25.53           152.50
1709.0
3                 0.09744  ...          26.50            98.87
567.7
4                 0.05883  ...          16.67           152.20
1575.0

   worst smoothness  worst compactness  worst concavity  worst concave
points  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

ANALYSIS

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso
from sklearn import metrics
import matplotlib.pyplot as plt
import numpy as np

df_x = df.drop(columns="target")
df_y = df[["target"]]

X_train, X_test, y_train, y_test = train_test_split(df_x, df_y,
test_size=0.33, random_state=42)

print("TRAIN")
results  = framework.scikit_linearreg_fit(X_train, y_train)
preds = framework.linearreg_pred(X_train, results["model"], binarize =
True)
print(metrics.classification_report(y_train, preds))
print("mse ", metrics.mean_squared_error(y_train, preds))

print("")
print("TEST")
preds = framework.linearreg_pred(X_test, results["model"], binarize =
True)
print(metrics.classification_report(y_test, preds))
print("mse ", metrics.mean_squared_error(y_test, preds))

print("")
print("BIAS & VARIANZA")
print(str(framework.scikit_bias_variance(df_x, df_y,
results["model"])))
```

```
TRAIN
              precision    recall  f1-score   support

           0       1.00      0.92      0.96       145
           1       0.95      1.00      0.98       236

    accuracy                           0.97       381
   macro avg       0.98      0.96      0.97       381
weighted avg       0.97      0.97      0.97       381

mse  0.031496062992125984

TEST
              precision    recall  f1-score   support

           0       0.95      0.88      0.91        67
           1       0.94      0.98      0.96       121

    accuracy                           0.94       188
   macro avg       0.94      0.93      0.94       188
weighted avg       0.94      0.94      0.94       188

mse  0.05851063829787234

BIAS & VARIANZA
{'mse_bias': 0.07485001189730058, 'bias': 0.06635100406158186, 'var':
0.008499007835718723}
```

LOW BIAS LOW VAR

meaning it is balanced

REGULARIZDORES

```
print("")
print("REGULARIZADORES")
results = framework.ridge_lasso(X_train, y_train)
print(str(results))
```

```
REGULARIZADORES
{'ridge coefficients': array([ 8.20188311e-02,  1.79122546e-03, -
4.14369479e-04, -7.33499284e-04,
       -2.48301115e-01,  3.20316789e-01, -3.42232429e-01, -
6.10629652e-01,
       -8.31205126e-02,  2.44714622e-02, -4.04040617e-01,
2.78459933e-02,
       -1.05465839e-02,  1.86169037e-03, -1.37577520e-01,
1.57978223e-01,
        5.66692414e-01, -7.04997717e-02, -6.13878521e-02,
2.57295617e-02,
       -1.92070436e-01, -1.38262629e-02,  5.38238852e-03,
9.64290796e-04,
       -5.81526588e-01, -3.61138099e-02, -4.67191258e-01, -
6.35274565e-01,
       -4.69675439e-01, -4.09294437e-02]), 'ridge intercept':
2.5512235360549083, 'ridge model': Ridge(alpha=0.3), 'lasso
coefficients': array([-0.        , -0.        , -0.        ,
0.00025427, -0.        ,
       -0.        , -0.        , -0.        , -0.        , -
0.        ,
       -0.        , -0.        , -0.        ,  0.        , -
0.        ,
       -0.        , -0.        , -0.        , -0.        , -
0.        ,
       -0.        , -0.00561035, -0.01806095,  0.00028739, -
0.        ,
       -0.        , -0.        , -0.        , -0.        , -
0.        ]), 'lasso intercept': 2.2816762262622228, 'lasso model':
Lasso(alpha=0.3)}
```

```
print("TRAIN RISSO")
preds = framework.linearreg_pred(X_train, results["ridge model"],
binarize = True)
print(metrics.classification_report(y_train, preds))
print("mse ", metrics.mean_squared_error(y_train, preds))
print("TEST RISSO")
preds = framework.linearreg_pred(X_test, results["ridge model"],
binarize = True)
print(metrics.classification_report(y_test, preds))
print("mse ", metrics.mean_squared_error(y_test, preds))
```

```
TRAIN RISSO
              precision    recall  f1-score   support

           0       0.99      0.89      0.94       145
           1       0.94      1.00      0.97       236

    accuracy                           0.96       381
   macro avg       0.96      0.94      0.95       381
weighted avg       0.96      0.96      0.95       381

mse  0.04461942257217848
TEST RISSO
              precision    recall  f1-score   support

           0       0.97      0.96      0.96        67
           1       0.98      0.98      0.98       121

    accuracy                           0.97       188
   macro avg       0.97      0.97      0.97       188
weighted avg       0.97      0.97      0.97       188

mse  0.026595744680851064
```

```
print("TRAIN LASSO")
preds = framework.linearreg_pred(X_train, results["lasso model"],
binarize = True)
print(metrics.classification_report(y_train, preds))
print("mse ", metrics.mean_squared_error(y_train, preds))
print("TEST LASSO")
preds = framework.linearreg_pred(X_test, results["ridge model"],
binarize = True)
print(metrics.classification_report(y_test, preds))
print("mse ", metrics.mean_squared_error(y_test, preds))
```

```
TRAIN LASSO
              precision    recall  f1-score   support

           0       0.97      0.83      0.89       145
           1       0.90      0.98      0.94       236

    accuracy                           0.92       381
   macro avg       0.94      0.91      0.92       381
weighted avg       0.93      0.92      0.92       381

mse  0.07611548556430446
TEST LASSO
              precision    recall  f1-score   support

           0       0.97      0.96      0.96        67
           1       0.98      0.98      0.98       121

    accuracy                           0.97       188
   macro avg       0.97      0.97      0.97       188
weighted avg       0.97      0.97      0.97       188

mse  0.026595744680851064
```

Los regularizadores si mejoraron la predicción de .03 decimales aprox