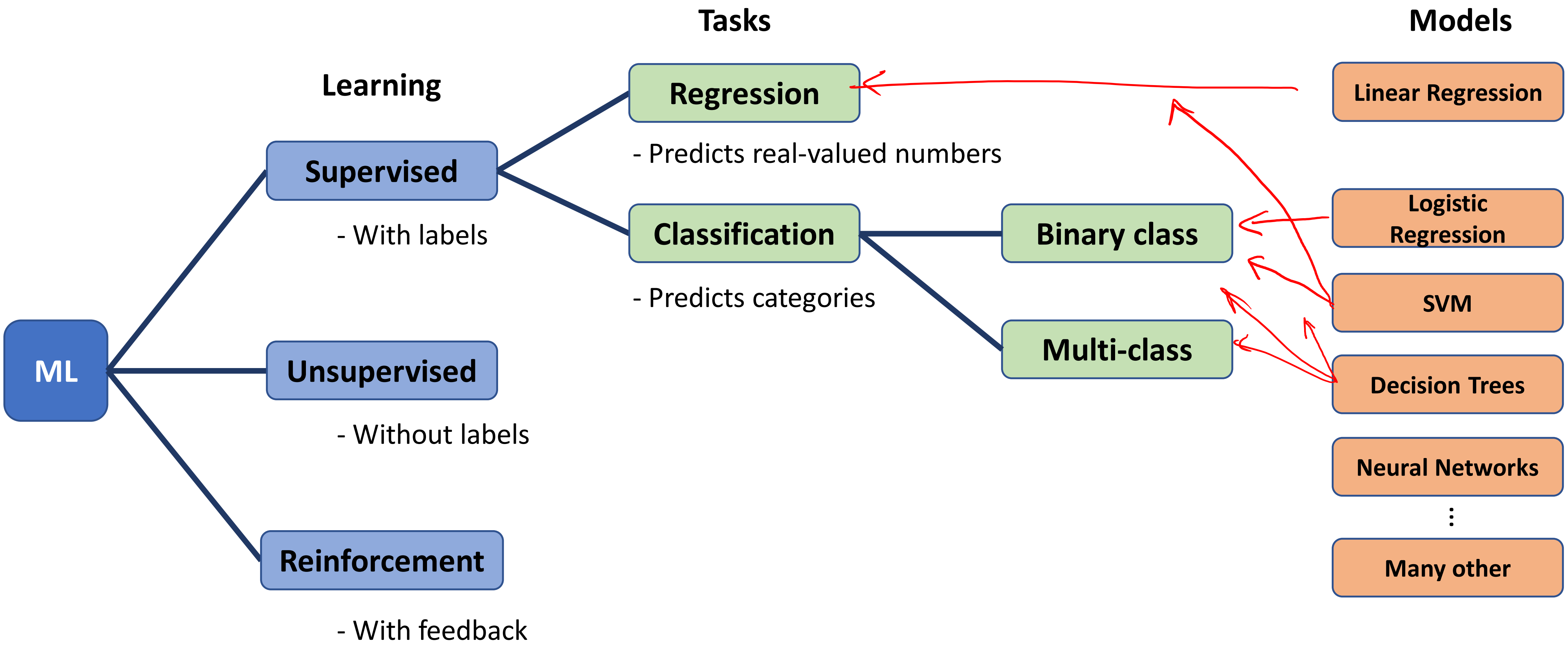
Logistic Regression

Introduction

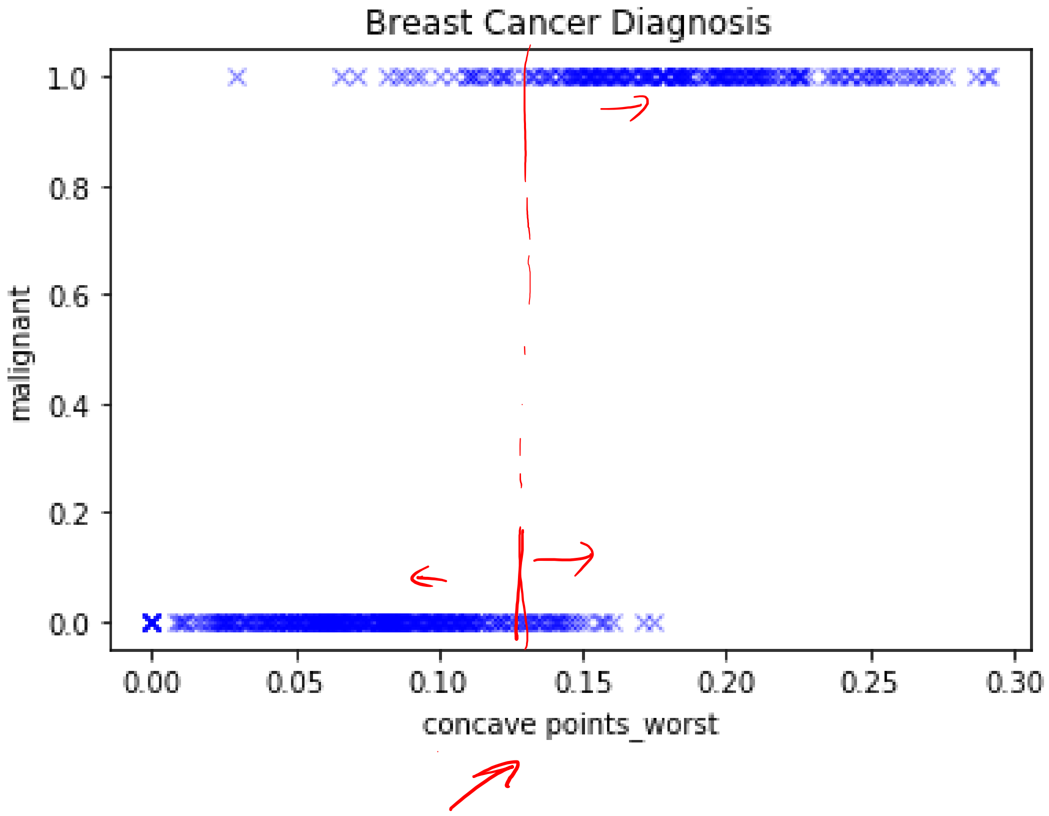
Review- types of machine learning problems



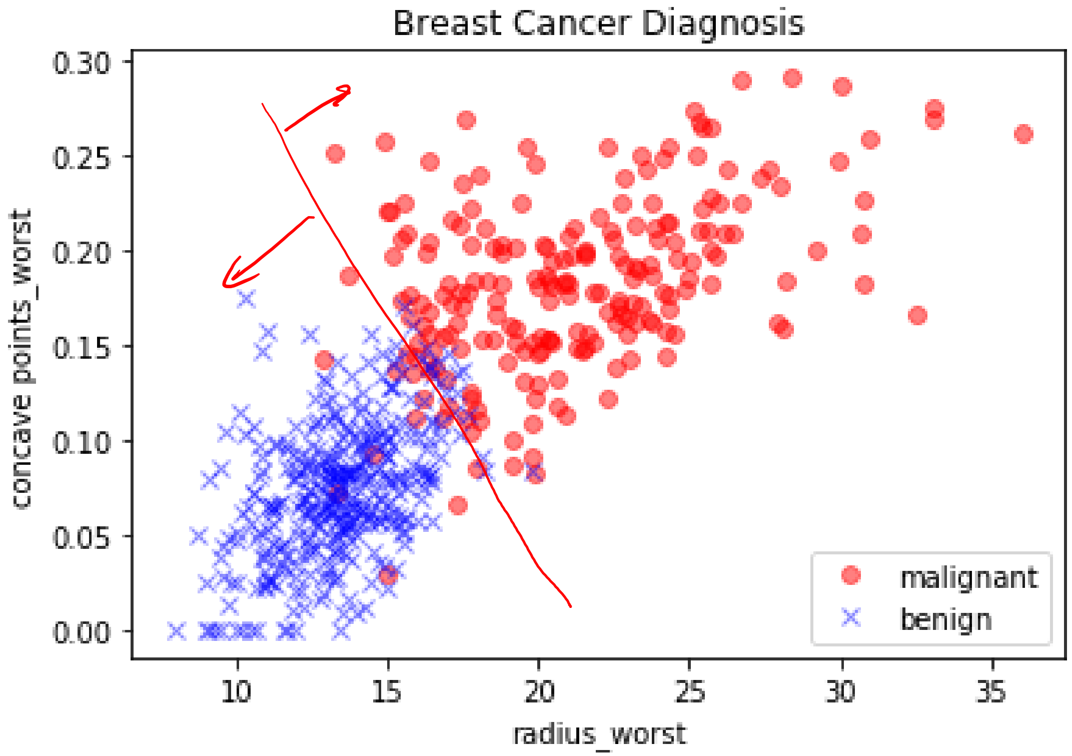
Binary Classification

Yes or No problem

- Creditcard Default
- Fradulant Insurance Claim
- Spam Filtering
- Medical Diagnosis
- Survival Prediction
- Customer Retention
- Image Recognition
- Sentiment Analysis



Breast Cancer Diagnosis
malignant
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
concave points_worst



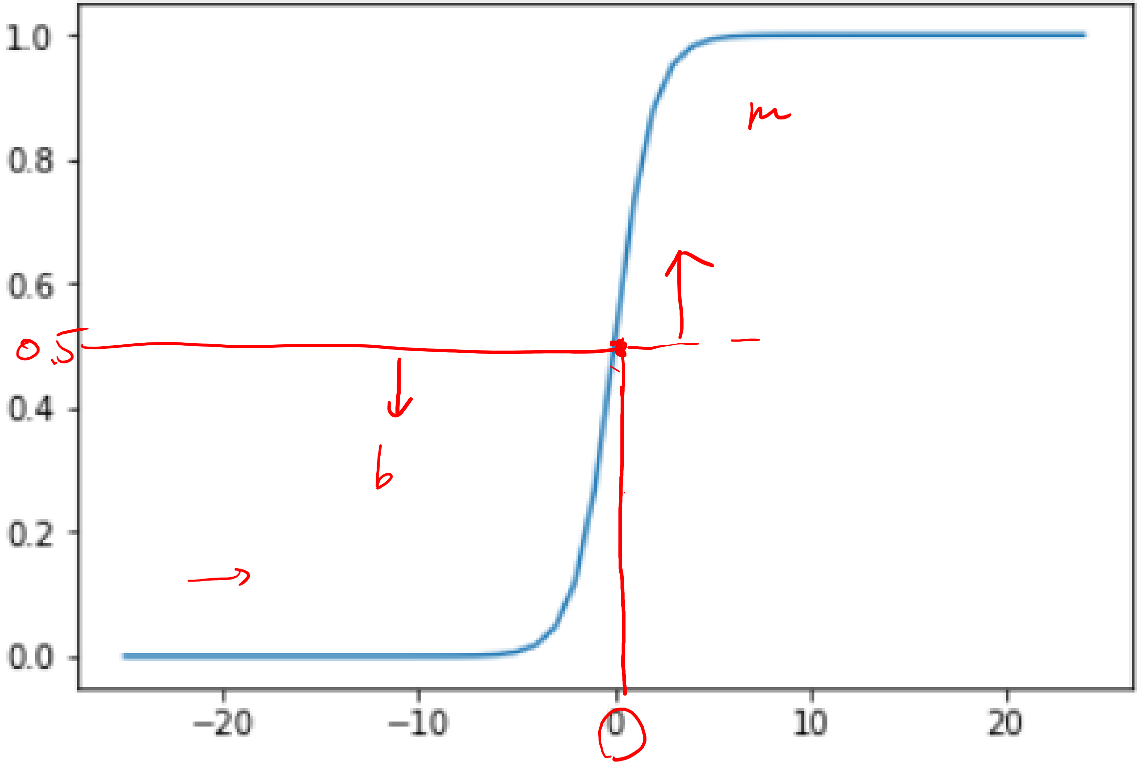
Breast Cancer Diagnosis
concave points_worst
0.30
0.25
0.20
0.15
0.10
0.05
0.00
10
15
20
25
30
35
radius_worst
malignant
benign

Logistic Function

$$P^{(i)} \in \mathbb{R}[0,1]$$

$$P^{(i)} = \sigma(z^{(i)})$$

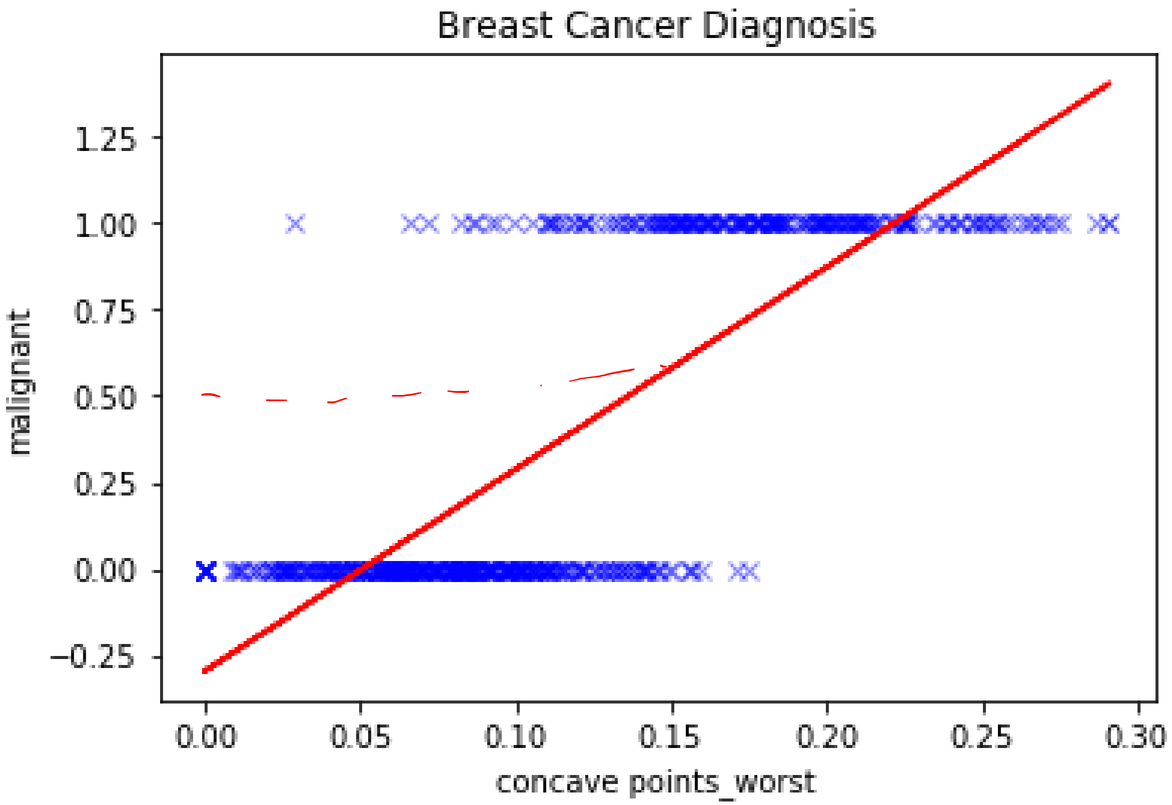
$$\boxed{\sigma(z)} = \frac{1}{1+e^{-z}} = \frac{1}{2}$$



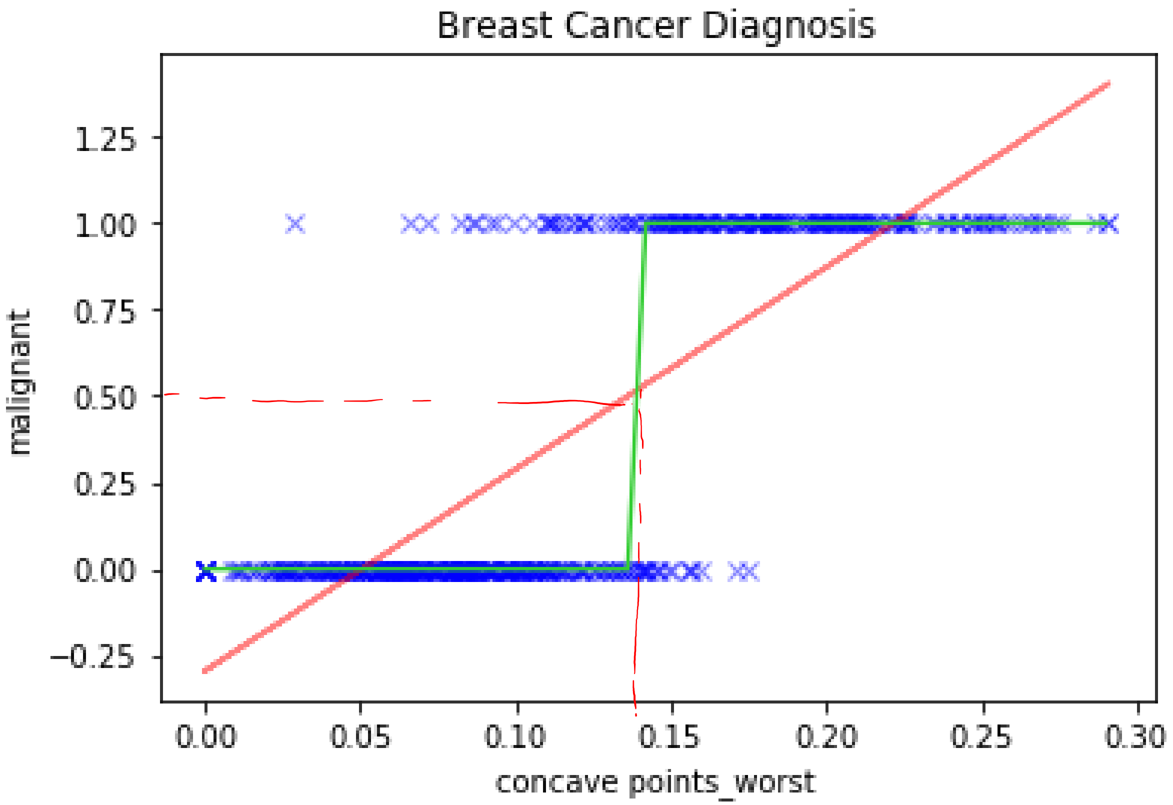
$$z^{(i)} = \boldsymbol{W} \cdot \boldsymbol{X} + b = 0$$

Called "logit" and is related to the decision boundary

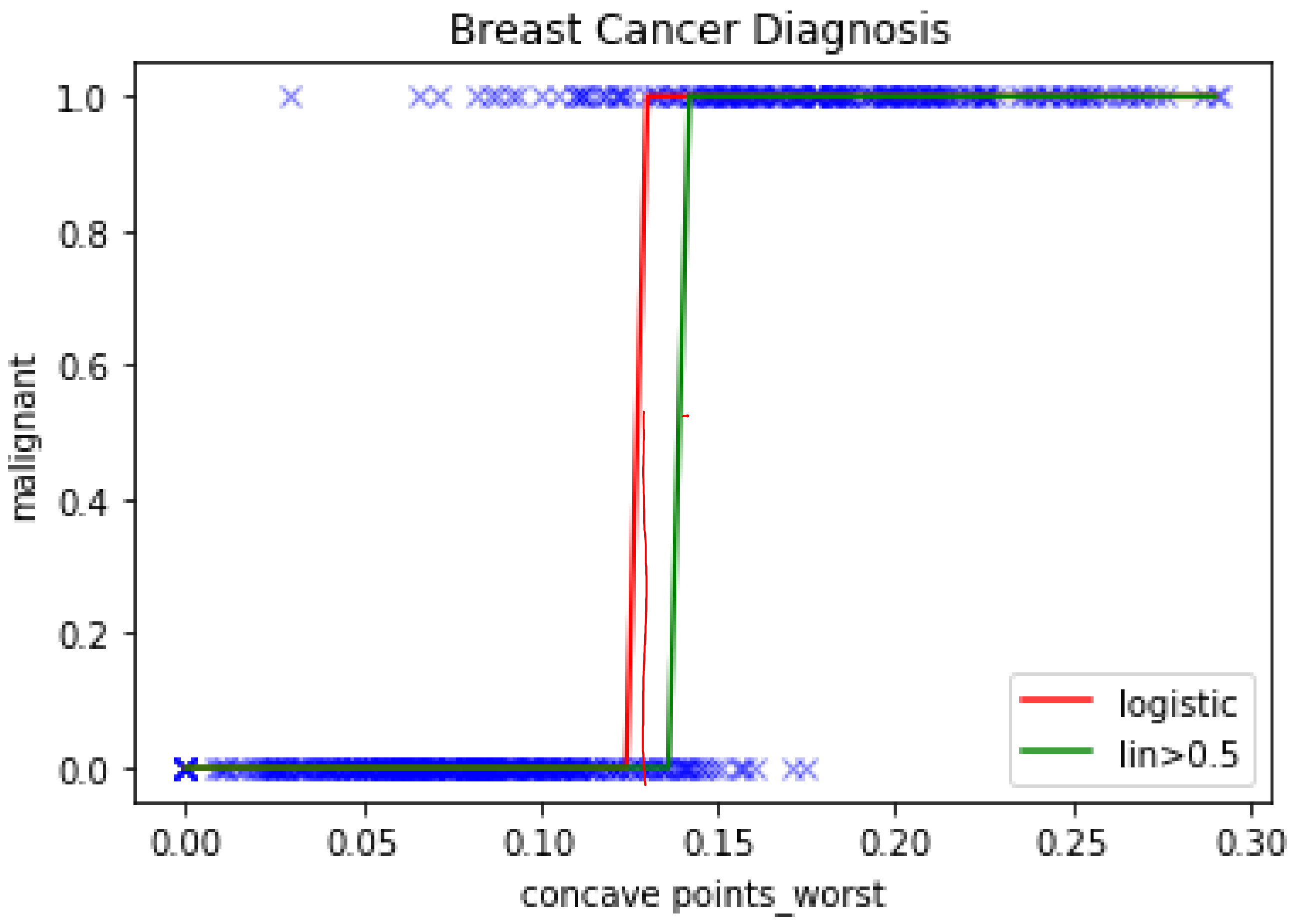
Logistic function as probability



linreg.predict(x)



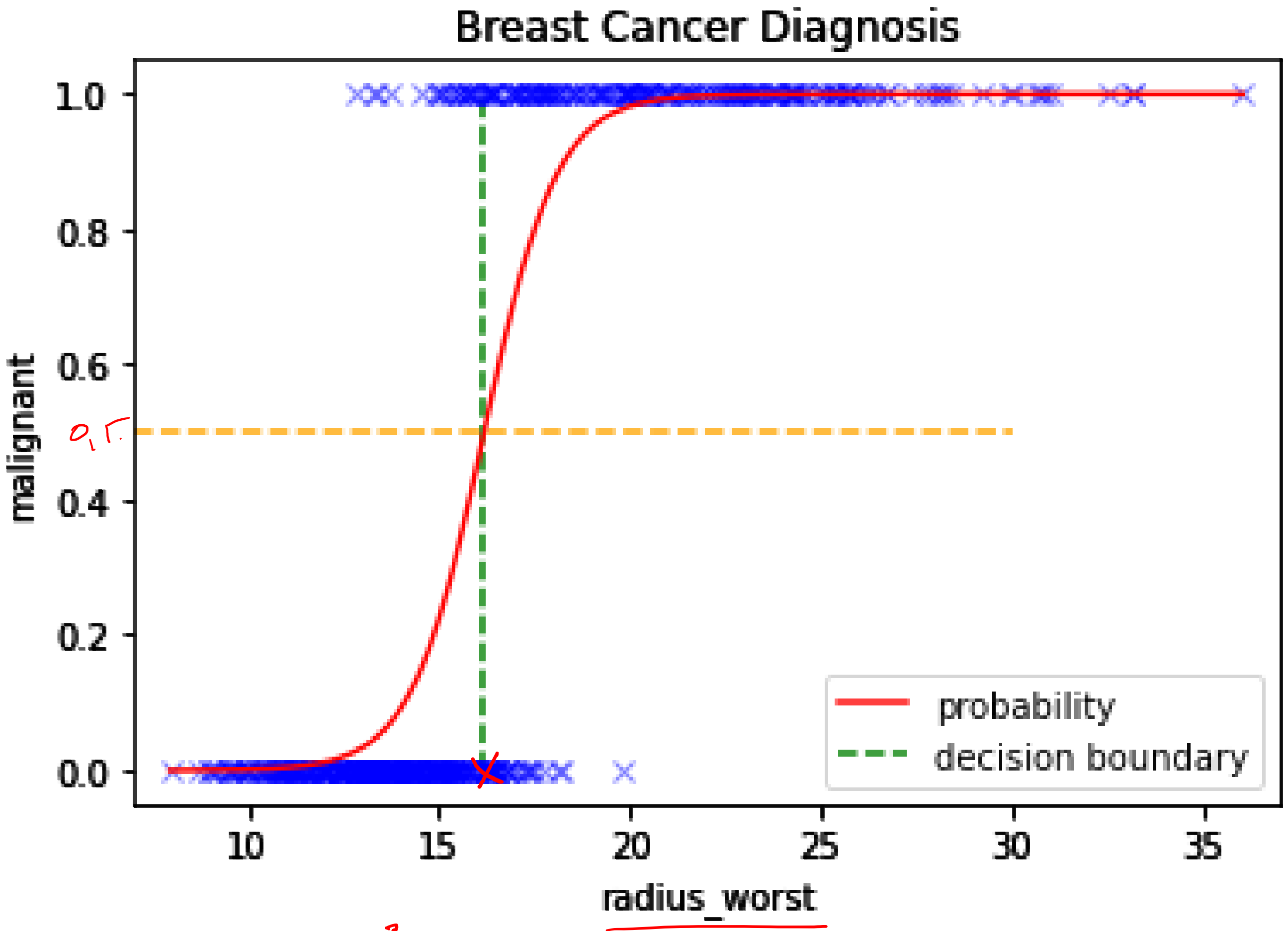
linreg.predict(x) > 0.5



Breast Cancer Diagnosis
1.0
0.8
0.6
0.4
0.2
0.0
malignant
0.00
0.05
0.10
0.15
0.20
0.25
0.30
concave points_worst
logistic
lin>0.5

Decision boundary- Univariate

radius_worst	label
13.05	0
16.39	1
10.85	0
21.86	1
21.31	1

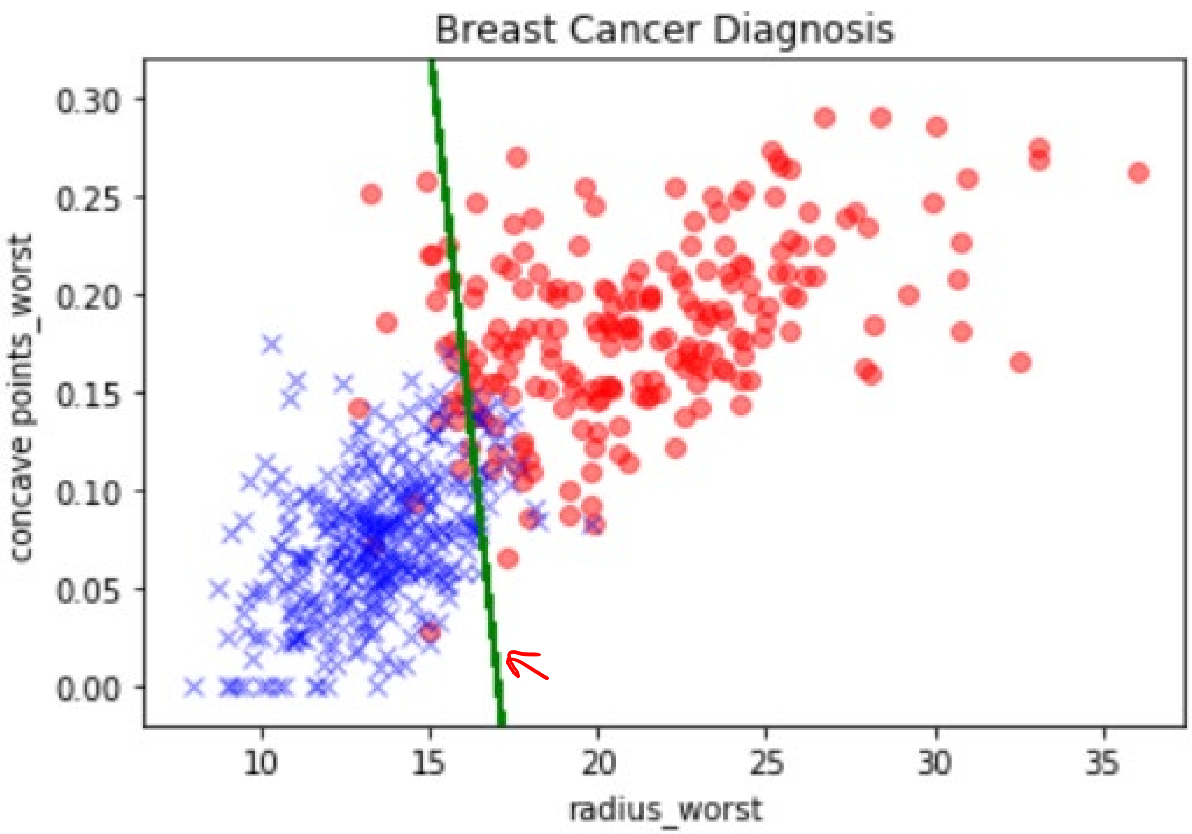


$z = 1.05\ X - 16.99 = 0$

Decision boundary- Multivariate

Breast Cancer Diagnosis

radius_worst	concave points_worst	label
13.05	0.08263	0
16.39	0.16730	1
10.85	0.14650	0
21.86	0.15100	1
21.31	0.14900	1



z= -16.7 + 1.02 x1+ 2.13 x2=0

What if we have multiple categories?

Logistic

A B C

LR1 A !A

LR2 B vs !B

LR3 C vs !C

one versus Rest

Softmax = multinomial LR

Logit

$$z^{(i)} = \boldsymbol{W} \cdot \boldsymbol{X}^{(i)} + b = \sum_j^p W_j X_j^{(i)} + b$$

$$z_k^{(i)} = \boldsymbol{W}_k \cdot \boldsymbol{X}^{(i)} + b = \sum_j^p W_{jk} X_j^{(i)} + b$$

category

Probability

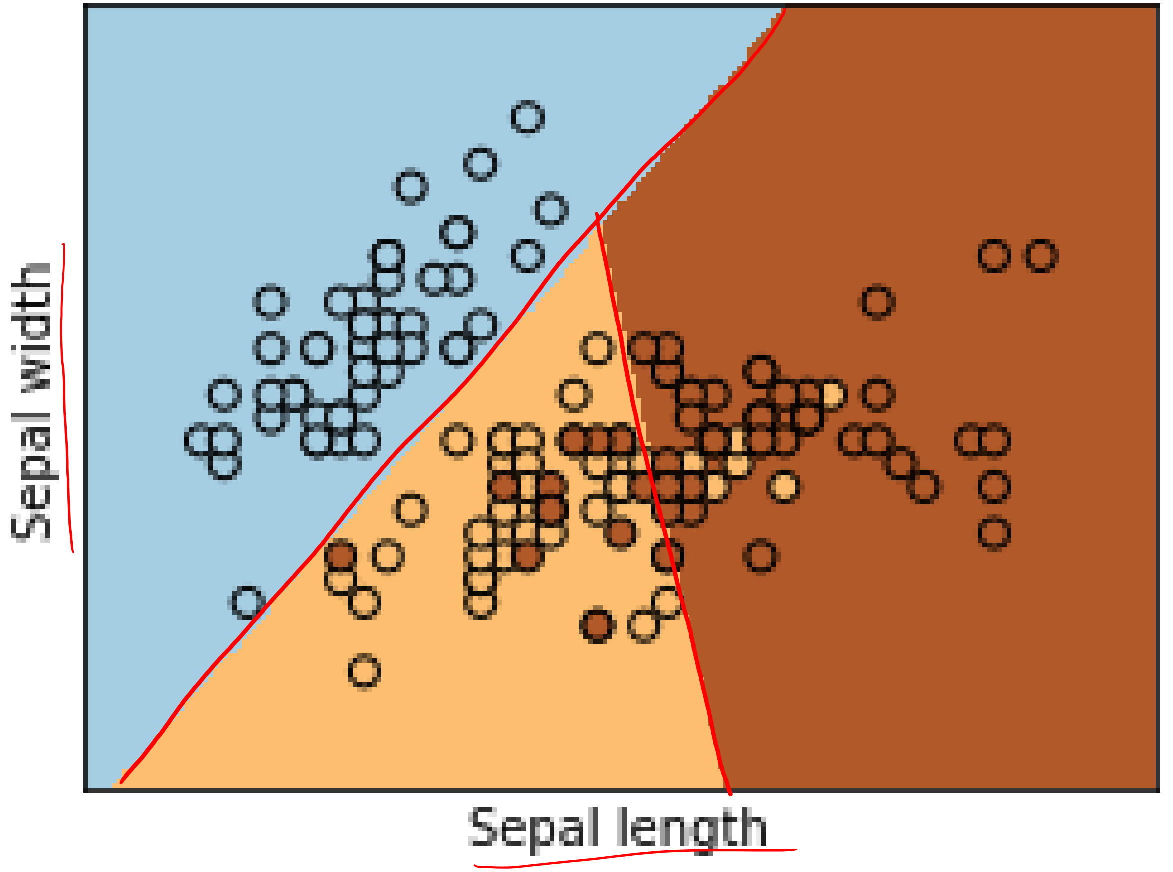
$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_{k'}^{K} e^{z_{k'}}}$$

#1 $P_A + P_B + P_C = 1$

A B C → multi label problem

Softmax regression



https://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html

Logistic Regression

Optimization

Estimating parameters in logistic regression

$P(y=1|x)_{w,b} = \sigma(z) = \frac{1}{1+e^{-z}}$

$z = w \cdot x + b$

Maximum Likelihood

$$\max \log\left[\ell(\beta_0, \beta)\right] = \log\left[\prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))\right]$$

y_1, y_2, y_3, y_4, y_5

$1 \quad 1 \quad 0 \quad 0 \quad 1$

$x \rightarrow \boxed{LR} \rightarrow P$

$P_1 = 1 \quad P_2 = 1 \quad P_3 \quad P_4 \quad P_5 \ldots$

$$\boxed{P_1 \times P_2 \times (1-P_3) \times (1-P_4) \cdot P_5}$$

$$\mathcal{L}_{BCE} = -\sum_i \left(y_i \log P_i + (1-y_i) \cdot \log(1-P_i) \right)$$

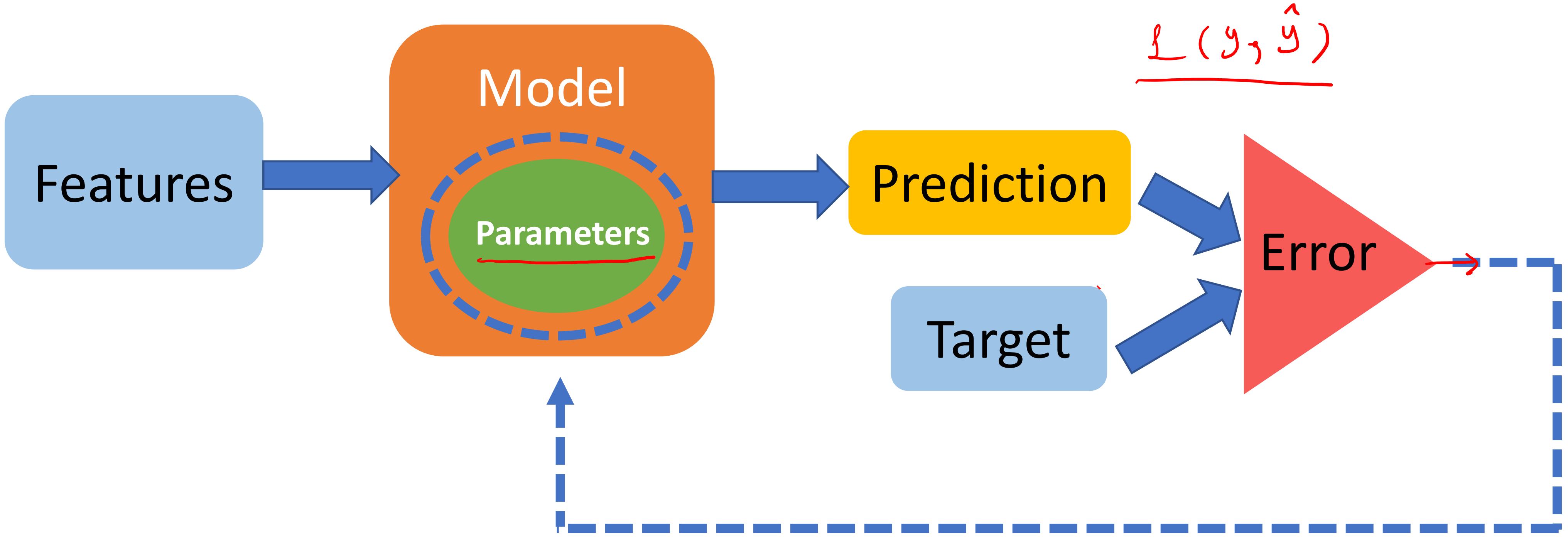
Estimating parameters in logistic regression

Cross Entropy

$$\mathcal{H}(P,Q) = -\sum_{ij} P_{ij} \log(Q_{ij}) \quad +$$

$$= -\frac{1}{m}\sum_{i}^{m} y_i \log \hat{p}_i + (1-y_i) \log (1-\hat{p}_i)$$

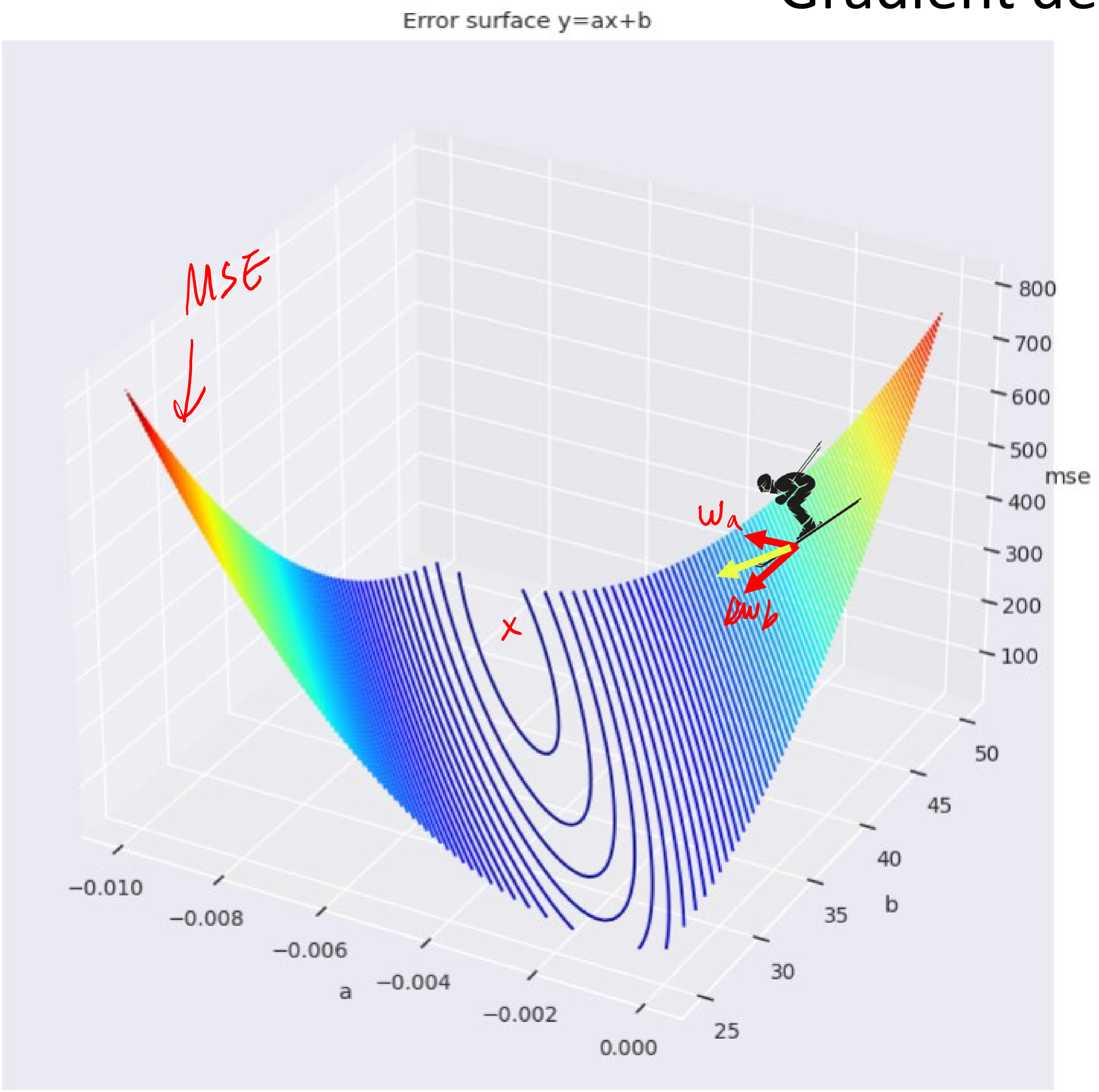
Searching Parameters



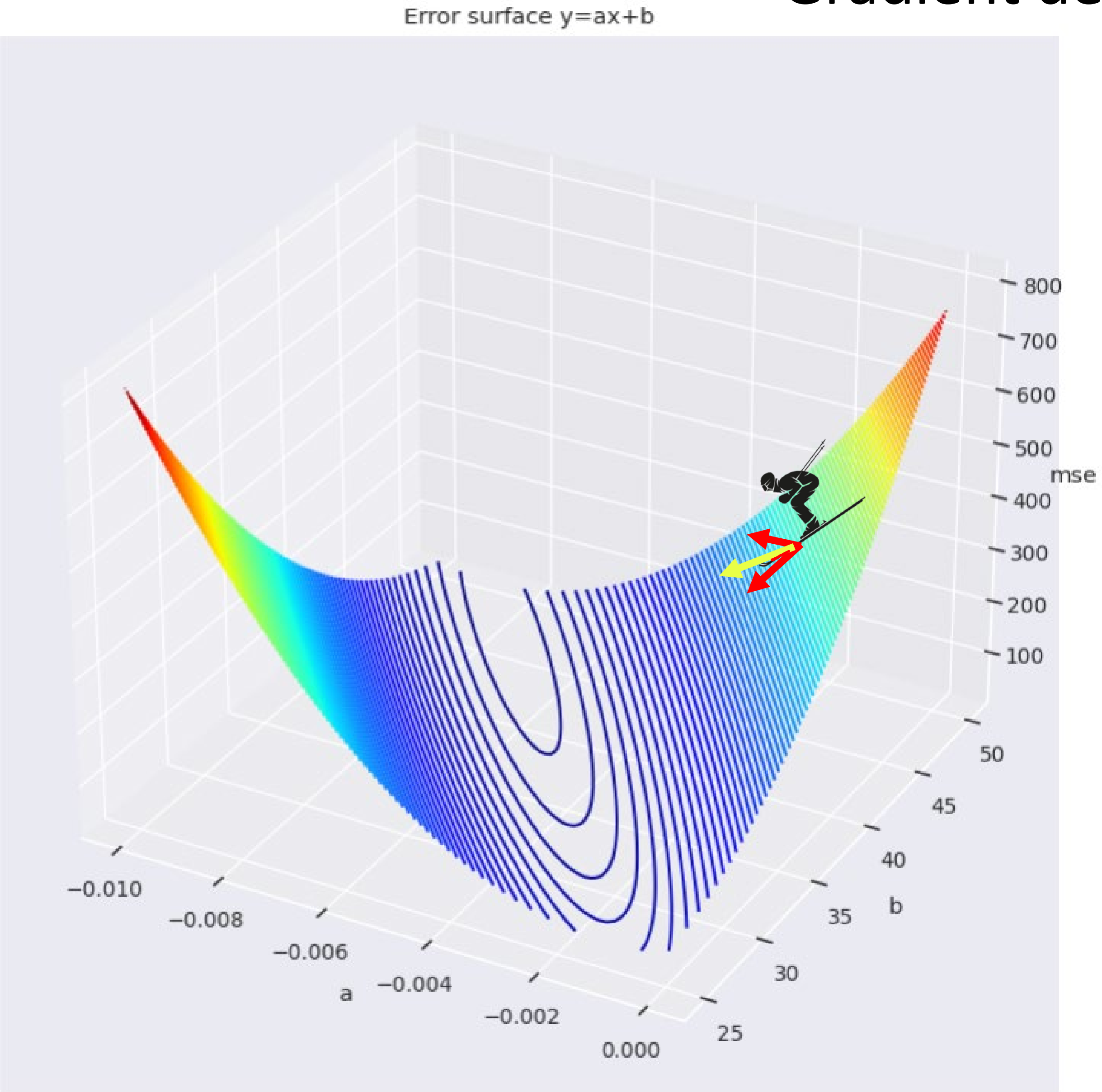
Parameter update

Gradient Descent

Gradient descent



Gradient descent (MSE loss)



Loss function

$$L = \frac{1}{2n}\sum_i^n (y_i - f(x_i))^2 = \frac{1}{2n}\sum_i^n (y_i - (ax_i + b))^2$$

$$\frac{\partial}{\partial a} f^2 = 2f \cdot \frac{\partial f}{\partial a}$$

Gradients

$$\frac{\partial}{\partial x} g(f(x))$$

$$\nabla_a L = \frac{\partial L}{\partial a} = -\frac{1}{n}\sum_i^n (y_i - (ax_i + b))x_i$$

$$\nabla_b L = \frac{\partial L}{\partial b} = -\frac{1}{n}\sum_i^n (y_i - (ax_i + b))$$

Parameter(weight) update rule

learning rate

$$\omega = \omega - \boxed{\alpha}\nabla_\omega L$$

a, b

$w_{\cdot} - \alpha \cdot$

Gradient descent (BCE loss)

$$\mathcal{L}_{BCE} = -\frac{1}{m}\sum_{i}^{m} y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$\log'(x) = \frac{1}{x}$

$\sigma' = \sigma(1-\sigma)$

$z = wx + b$

$$\boxed{\frac{\partial \mathcal{L}(\sigma)}{\partial w}} = \boxed{\frac{\partial \mathcal{L}}{\partial \sigma}} \cdot \frac{\partial \sigma}{\partial w} \rightarrow \boxed{\sigma(1-\sigma)\cdot x}$$

$$-\left[\frac{y}{\sigma} + \frac{(1-y)}{(1-\sigma)}(-1)\right] \times$$

$$\frac{\partial \mathcal{L}}{\partial w} = \left[-\frac{y}{\sigma} + \frac{(1-y)}{(1-\sigma)}\right] \cdot \sigma(1-\sigma)\cdot \underline{x}$$

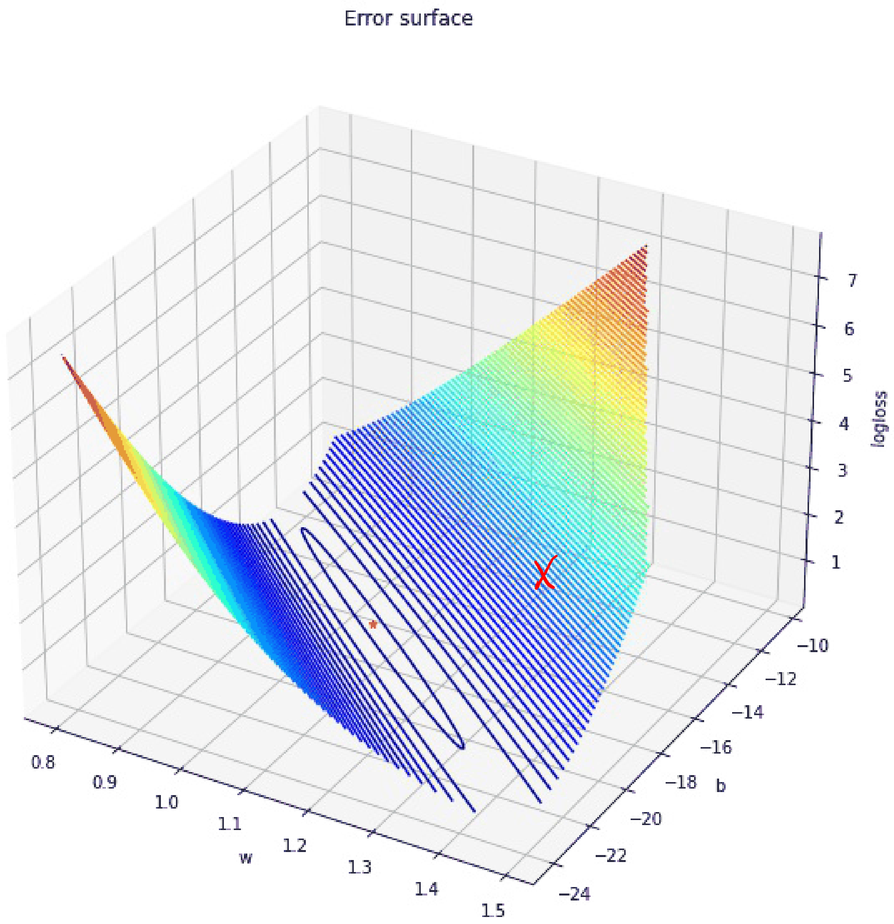
$$\frac{\partial \mathcal{L}}{\partial b} = [\quad " \quad] \quad " \; .$$

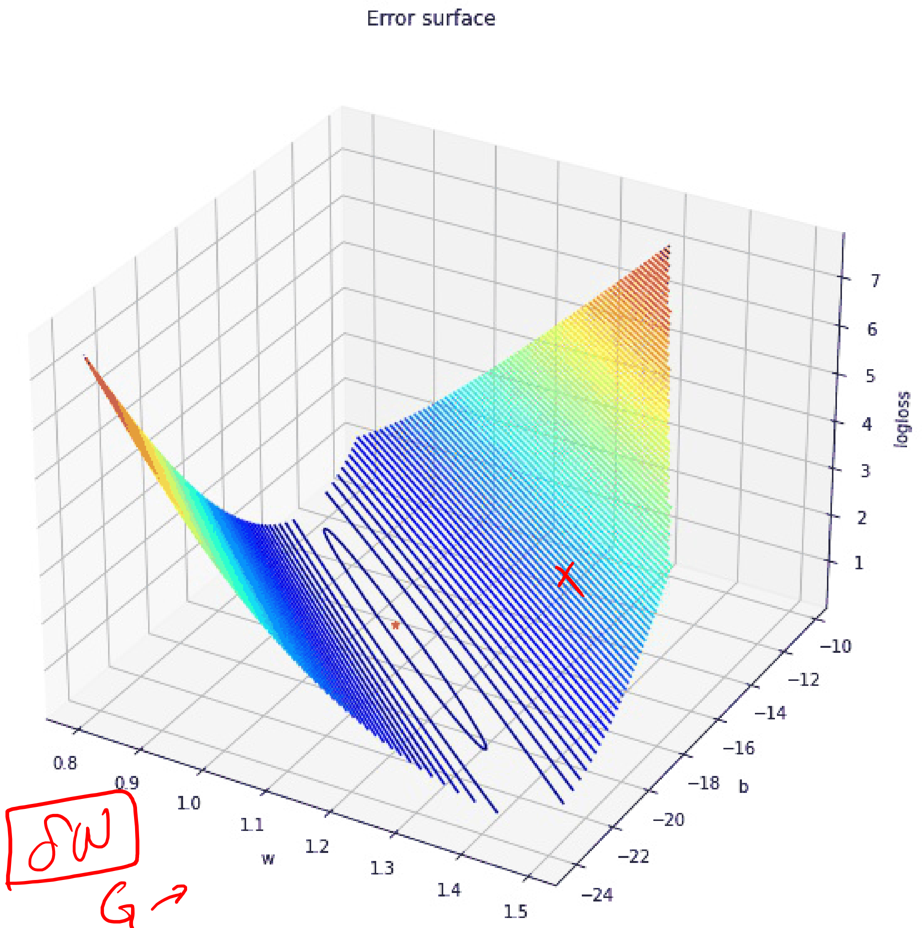
Newton's Method

Gradient descent $\quad w \leftarrow w - \alpha \nabla_w \mathcal{L}$ $\qquad O(n)\ ,\ O(n)$

Newton's Method $\quad w \leftarrow w - \alpha \dfrac{\nabla_w \mathcal{L}}{\boxed{\nabla_w^2 \mathcal{L}}} \rightarrow$ Hessian $\qquad \boxed{w \leftarrow w - \alpha \left[\dfrac{G}{H}\right]}$ $\qquad O(n^3)\ ,\ O(n^2)$

Gradient descent vs. Newton's method





$-\alpha G$

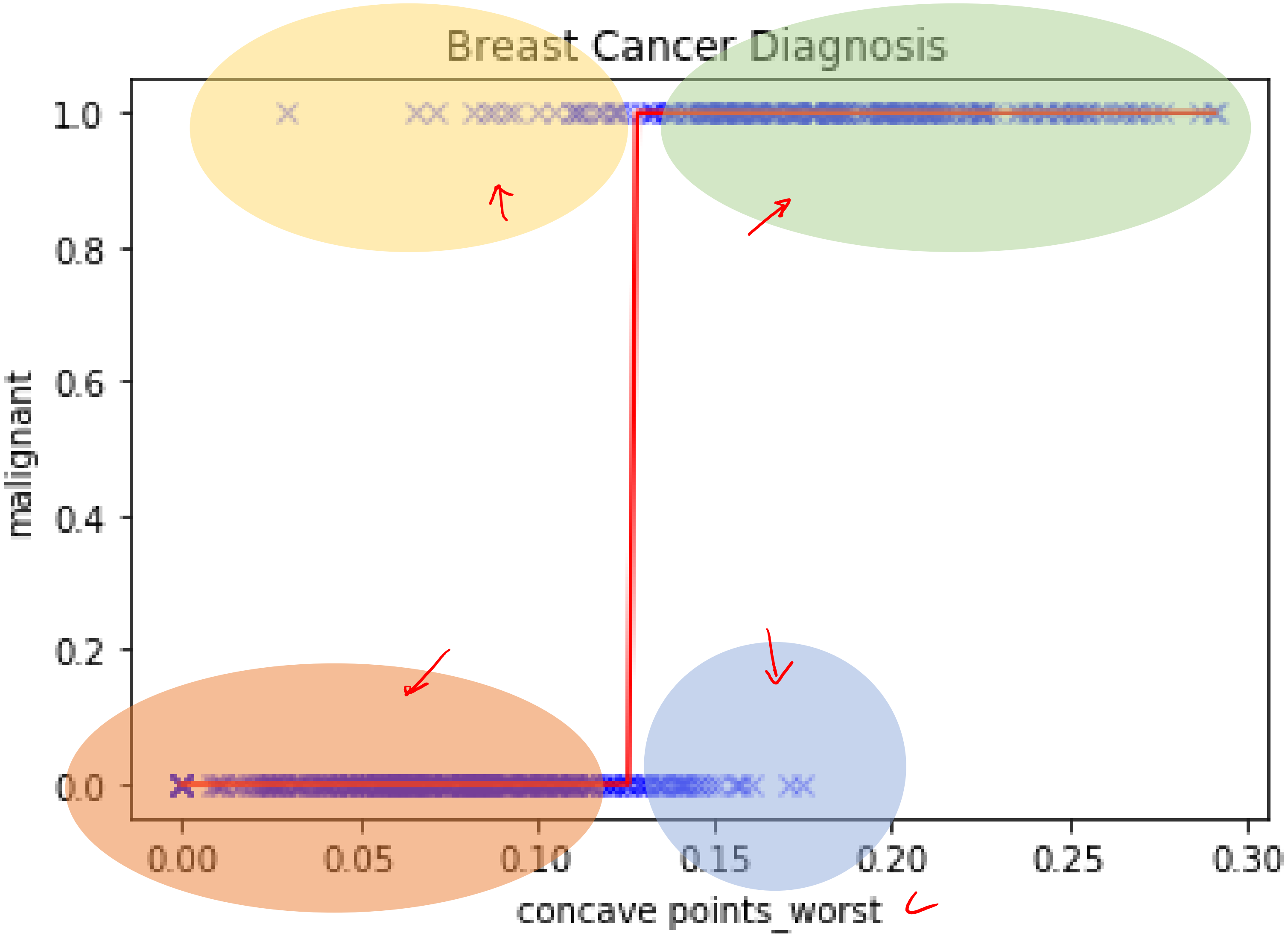
$\boxed{\delta W}$

$-\alpha \frac{\vec{G}}{\vec{H}}$

Logistic Regression

Performance Metrics

Interpreting Logistic Regression Result



Yt = 1, Yp = 1
True Positive

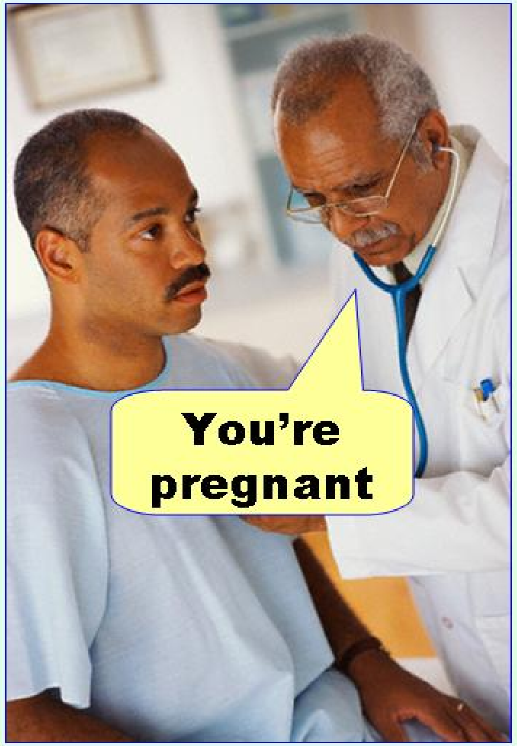
Yt = 0, Yp = 0
True Negative

Yt =0, Yp = 1
False Positive

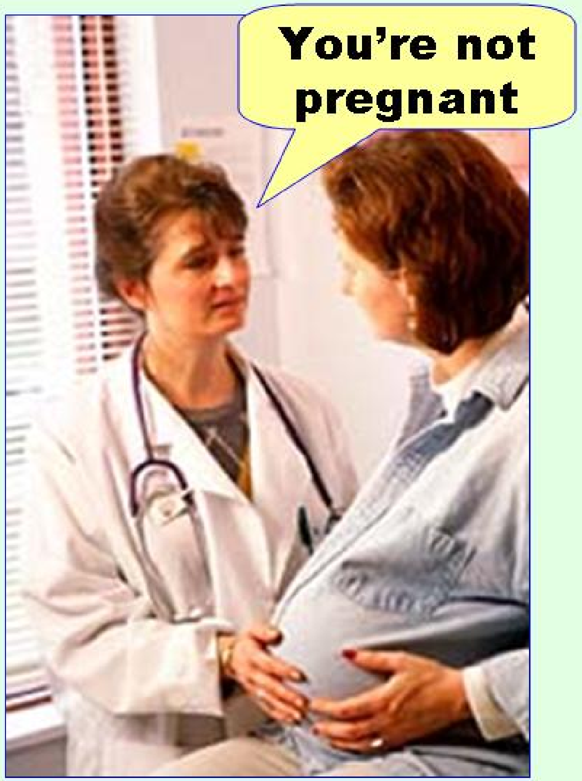
Yt = 1, Yp = 0
False Negative

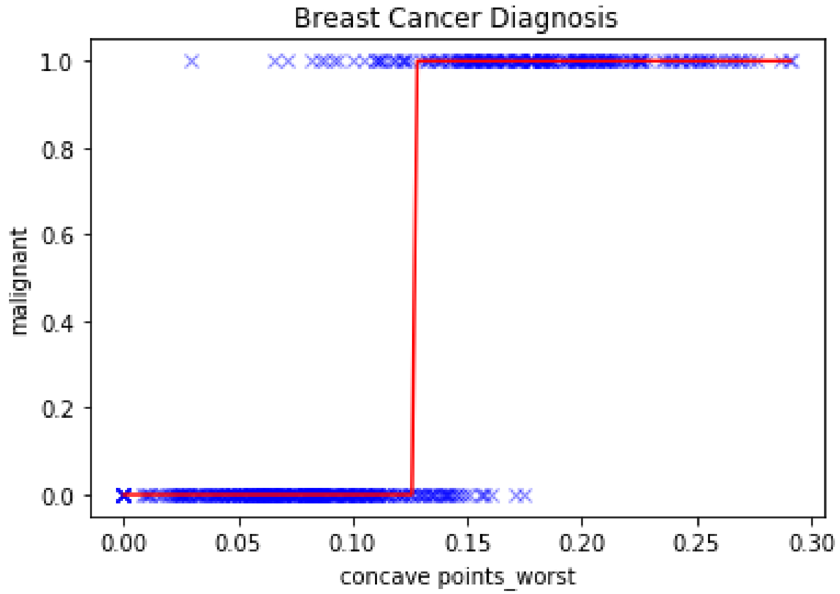
Type I error and Type II error

Type I error
(false positive)



Type II error
(false negative)





Binary Classification Performance Metrics

Confusion Matrix

TP, TN, FP, FN

Yt \ Yp	0	1
0	70 TN	1 FP
1	3 FN	40 TP

```python
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true, y_pred)
```

$$\mathrm{ACC} = \frac{TP+TN}{TP+TN+FP+FN}$$ Accuracy

$$\mathrm{TPR} = \frac{TP}{TP+FN} \rightarrow \frac{TP}{P(\text{data})}$$ Recall, Sensitivity

$$\mathrm{TNR} = \frac{TN}{TN+FP} \rightarrow \frac{TN}{N(\text{data})}$$ Specificity, Selectivity

$$\mathrm{PPV} = \frac{TP}{TP+FP} \rightarrow \frac{TP}{P(\text{prediction})}$$ Positive predictive value, Precision

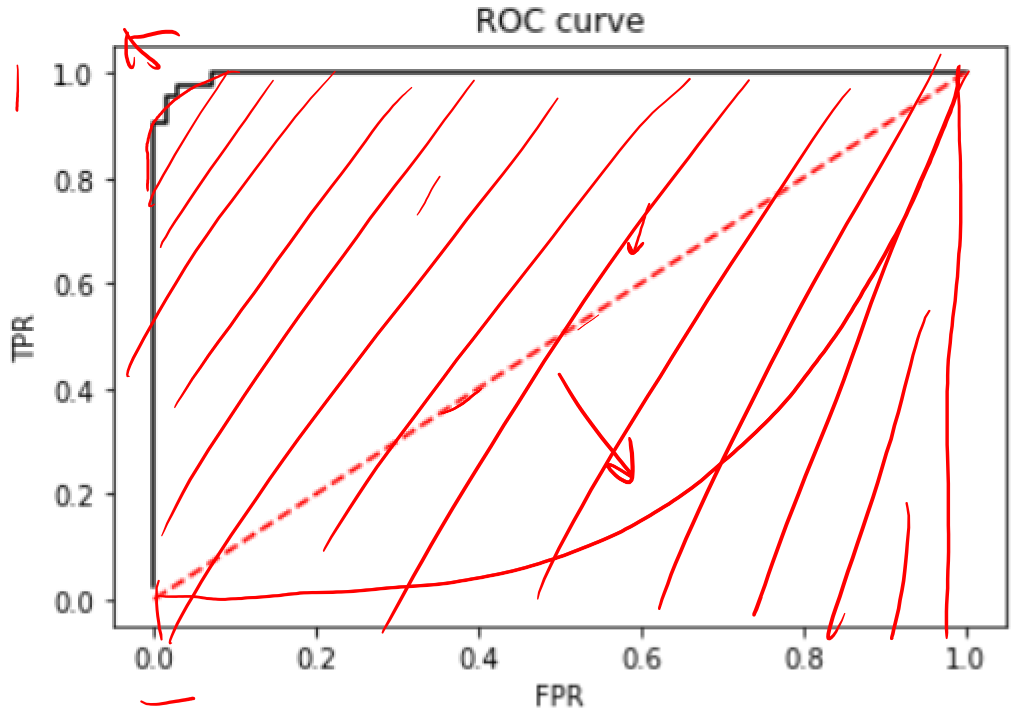
$$\mathrm{FPR} = \frac{FP}{FP+TN} \rightarrow \frac{FP}{N(\text{data})} = 1 - TNR$$ False-positive rate, Fall-out

$$\mathrm{FNR} = \frac{FN}{FN+TP} = \frac{FN}{P(\text{data})} = 1 - TPR$$ False-negative rate, Miss rate

$$\mathrm{F1} = \frac{2TP}{2TP+FP+FN} = \frac{2\,p \cdot r}{p+r}$$ Harmonic mean of Precision and recall

Performance Metrics- ROC, AUC

Receiver-Operating Characteristics Curve



AUC

0 – 1

AUC=0.99

Which Performance Metric should I choose?

99.9 % negative 99.9%
0.1 % positive
100 : negative

- Accuracy
- Sensitivity, Recall, TPR

$\frac{TP}{P}$ cancer FN

- Specificity, Sensitivity, TNR
- Precision, PPV
- False Positive Rate (fall-out) spam mail
- False Negative Rate (miss rate)
- F1 score
- AUC / ROC
- Confusion matrix

Why use Cross-Entropy, not Accuracy?

Cross Entropy

$$\mathcal{L}_{CE}(y,p) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{m} y_{i,j}\log \hat{p}_{i,j}$$

Accuracy $\frac{TP+TN}{ALL}$

A

Predicted probability			Target			Correct?
0.4	0.3	0.3	1	0	0	1
0.3	0.4	0.3	0	1	0	1
0.8	0.1	0.1	0	0	1	0

2/3

B

Predicted probability			Target			Correct?
0.8	0.1	0.1	1	0	0	1
0.1	0.8	0.1	0	1	0	1
0.4	0.3	0.3	0	0	1	0

2/3

Logistic Regression

library usage

Using sklearn's LogisticRegression

sklearn.linear_model.LogisticRegression

class sklearn.linear_model.LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

-1

solver : str, {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://github.com/scikit-learn/ (directory: scikit-learn/sklearn/linear_model/logistic.py)

Using sklearn's LogisticRegression

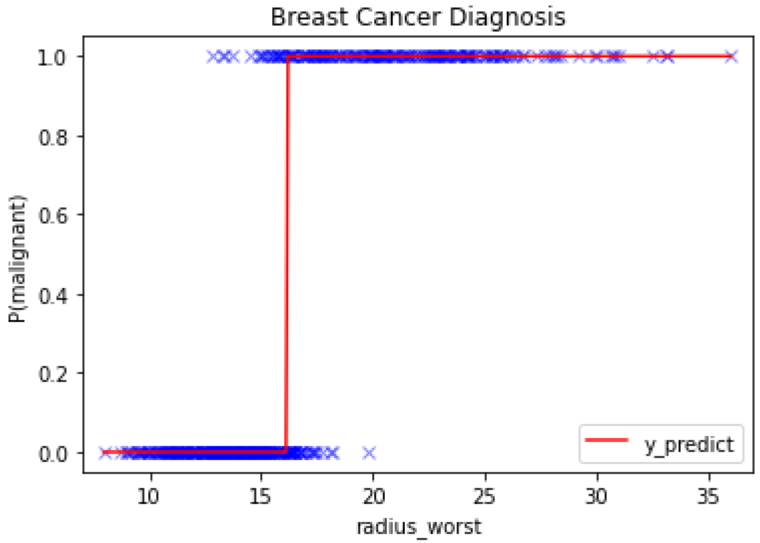
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression().fit(X, y)
```

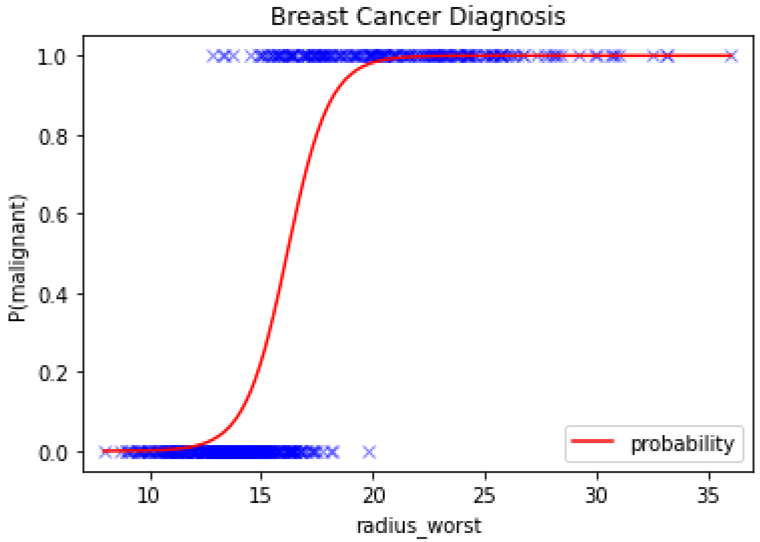
model.coef_

model.intercept_

y_p ← model.predict(X_test)

p ← model.predict_proba(X_test)





Using sklearn's LogisticRegression

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.linear_model import LogisticRegression as LR

clf = LR(class_weight="balanced", solver='liblinear').fit(X_train, y_train.ravel())
clf.score(X_test,y_test)
```

```
0.9649122807017544
```

```python
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score, recall_score

yp = clf.predict(X_test)
print('acc', accuracy_score(y_test, yp))
print('recall', recall_score(y_test, yp))
print('precision', precision_score(y_test, yp))
print('F1', f1_score(y_test, yp))
```

```
acc 0.9649122807017544
recall 0.9302325581395349
precision 0.975609756097561
F1 0.9523809523809524
```

```python
pd.DataFrame(confusion_matrix(y_test, yp, labels=[0,1]))
```

yp

yt

	0	1
0	70	1
1	3	40

Using sklearn's LogisticRegression

pre

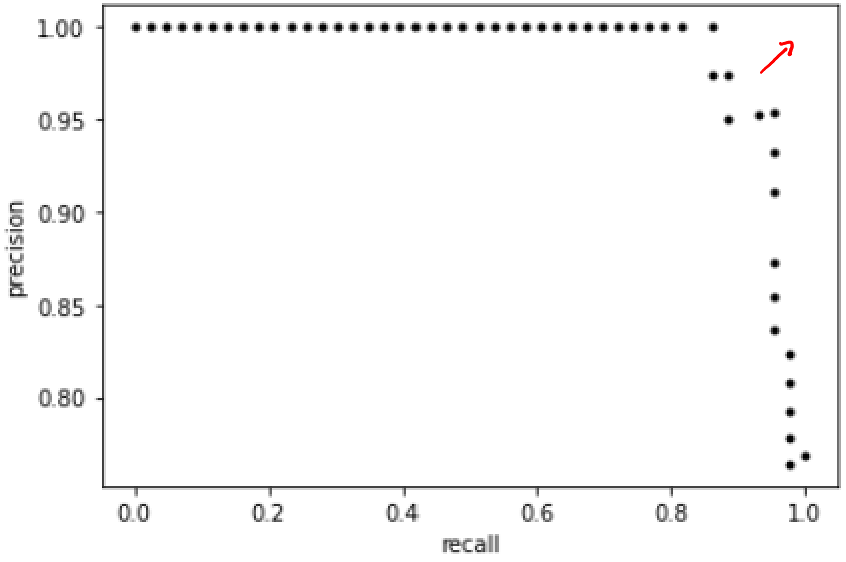
rec

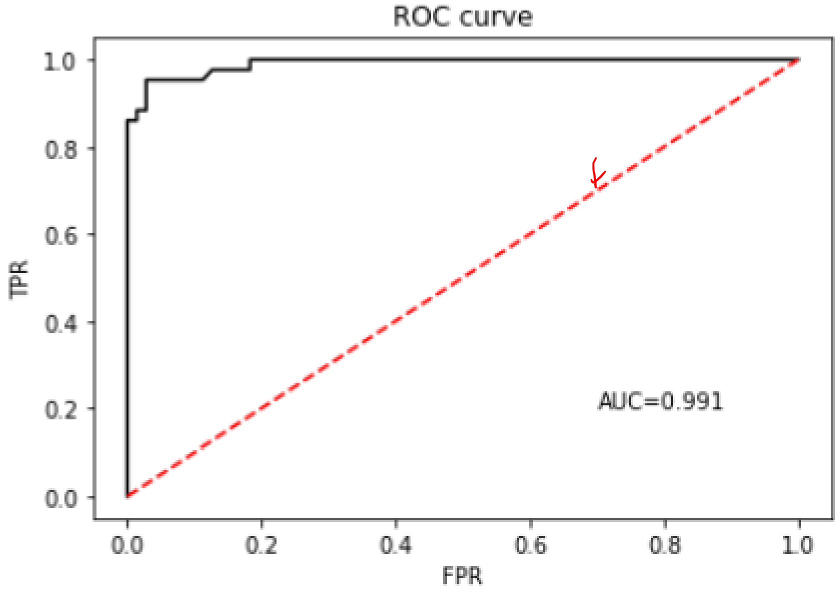
TPR

FPR

```
from sklearn.metrics import precision_recall_curve
ypp = clf.predict_proba(x_test)
pre, rec, th = precision_recall_curve(y_test,ypp[:,1])
plt.plot(rec,pre,'k.');
plt.ylabel('precision');
plt.xlabel('recall');
```

```
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, th = roc_curve(y_test,ypp[:,1])
auc = roc_auc_score(y_test,ypp[:,1])
plt.plot(fpr,tpr,'k-')
plt.plot(np.arange(0,1.1,0.1),np.arange(0,1.1,0.1),'r--')
plt.title('ROC curve');
plt.xlabel('FPR');
plt.ylabel('TPR');
plt.text(0.7,0.2,'AUC='+"{:.3f}".format(auc));
```





What about the statistics?

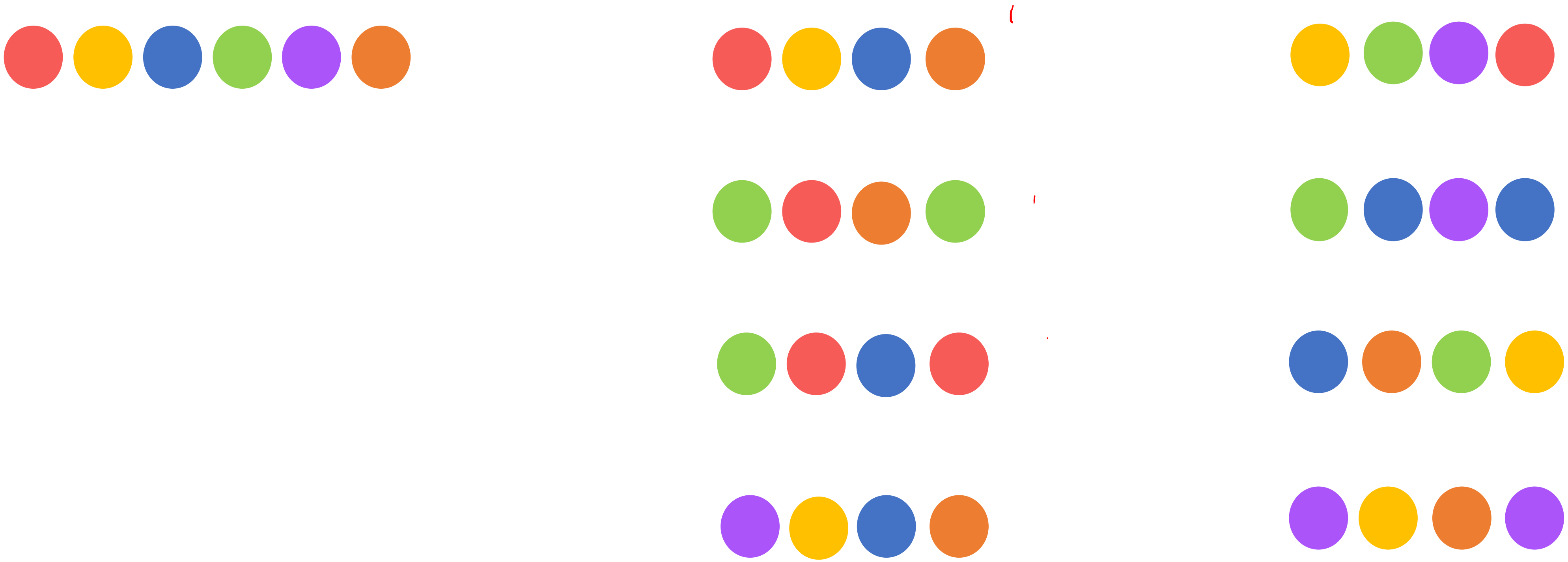
Statsmodels library

```
import statsmodels.api as sm
logit_model=sm.Logit(y_train,x_train)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.681033
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  455
Model:                          Logit   Df Residuals:                      454
Method:                           MLE   Df Model:                            0
Date:                Wed, 18 Sep 2019   Pseudo R-squ.:                -0.03232
Time:                        19:23:16   Log-Likelihood:                -309.87
converged:                       True   LL-Null:                       -300.17
                                        LLR p-value:                       nan
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1             2.3970      0.731      3.279      0.001       0.964       3.830
==============================================================================
```

What about the statistics?

Bootstrap (Resample)



What about the statistics?

sklearn.ensemble.BaggingClassifier

class `sklearn.ensemble.BaggingClassifier`(*base_estimator=None, n_estimators=10, *, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0*)

```
clf = BaggingClassifier(base_estimator=LogisticRegression(class_weight="balanced"),n_estimators=1000).fit(X,y)
```

```
clf.estimators_
```

```
[LogisticRegression(class_weight='balanced', random_state=1952926171),
 LogisticRegression(class_weight='balanced', random_state=1761383086),
 LogisticRegression(class_weight='balanced', random_state=1449071958),
 LogisticRegression(class_weight='balanced', random_state=1910541088),
 LogisticRegression(class_weight='balanced', random_state=1341730541),
```

```
clf.estimators_[0].coef_
```

```
array([[1.10855636, 1.6176168 ]])
```

```
clf.estimators_[0].intercept_
```

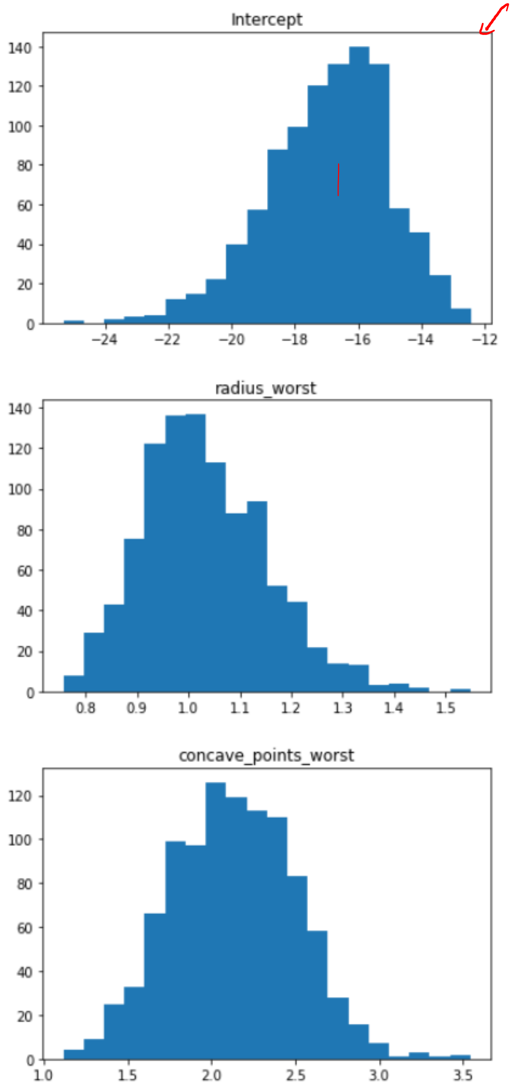
```
array([-18.31031081])
```

```
est=clf.estimators_[0]
list(est.intercept_)+list(est.coef_[0])
```

```
[-18.310310809003678, 1.1085563574633186, 1.6176167999143638]
```

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

What about the statistics?



scipy.stats.ttest_1samp

scipy.stats.ttest_1samp(*a, popmean, axis=0, nan_policy='propagate', alternative='two-sided'*)

```
t = ttest_1samp(coefs[:,i],0)
print(t)
t.statistic
```

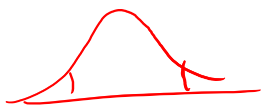
```
Ttest_1sampResult(statistic=181.38175127406623, pvalue=0.0)

181.38175127406623
```

coef	t-statistic	p-value
Intercept	-278.342110	0.0
radius_worst	270.601870	0.0
concave_points_worst	181.381751	0.0

$H_0 \rightarrow c = 0$
$H_1 \Rightarrow c \neq 0$

5%

$p < 0.025$



https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html