CSPB 3202 Artificial Intelligence

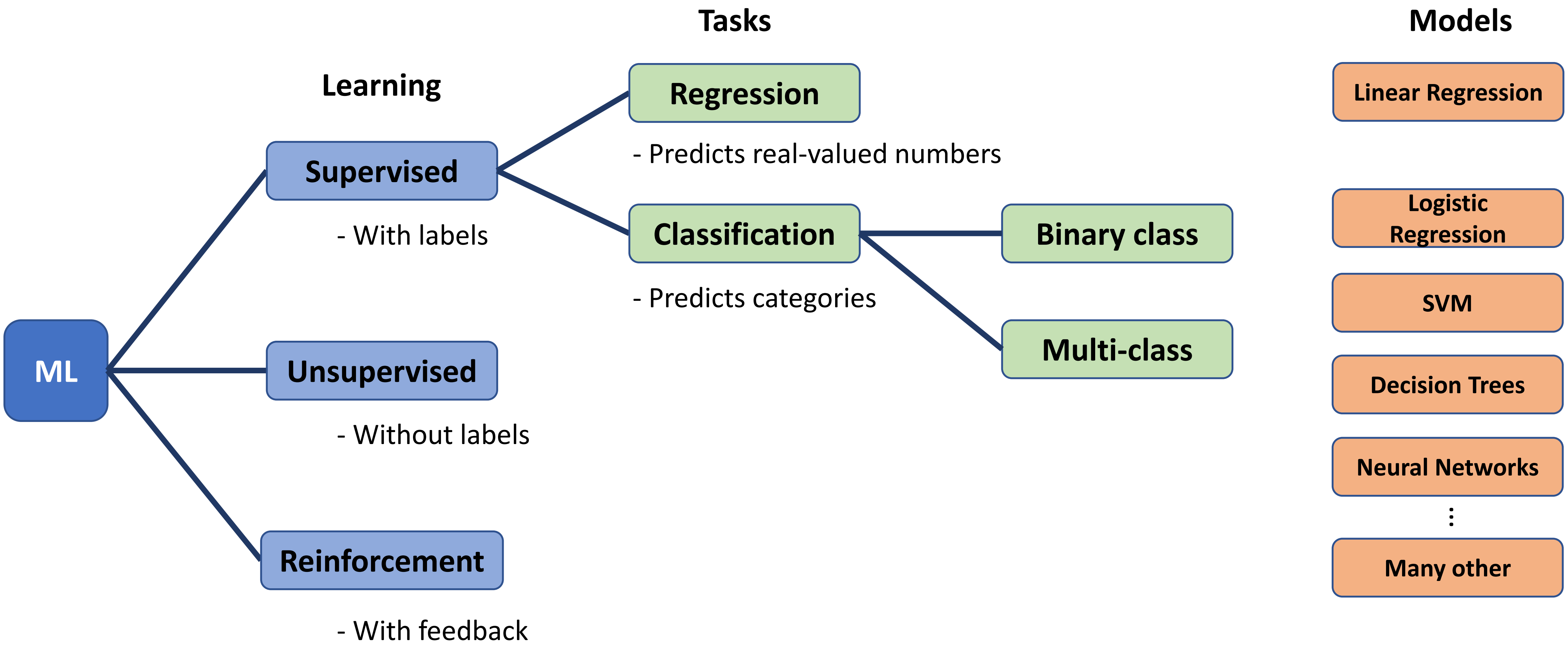
Machine Learning

Geena Kim

Logistic Regression

*Slide contents adopted from ISLR material

Review- types of machine learning problems



Review- Linear Regression

$$\hat{y}^{(i)} = \mathbf{w} \cdot \mathbf{x}^{(i)} + b$$

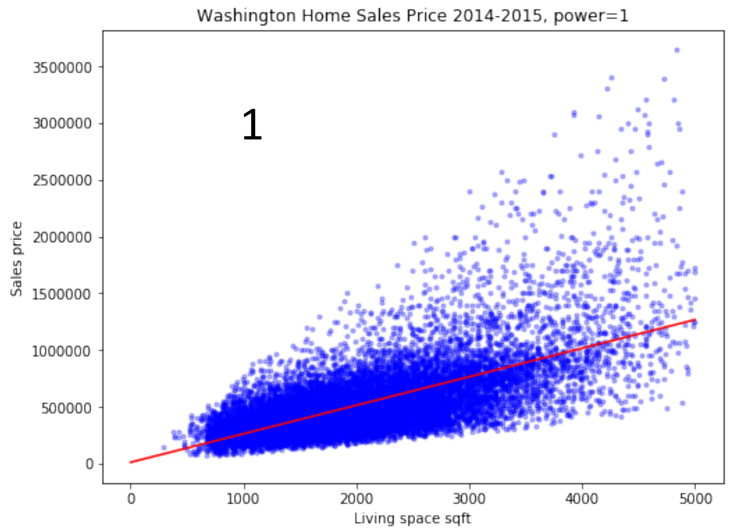
$$\mathbf{x} = \begin{array}{cc} w_1 & \\ w_2 & \\ & \\ & \end{array}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

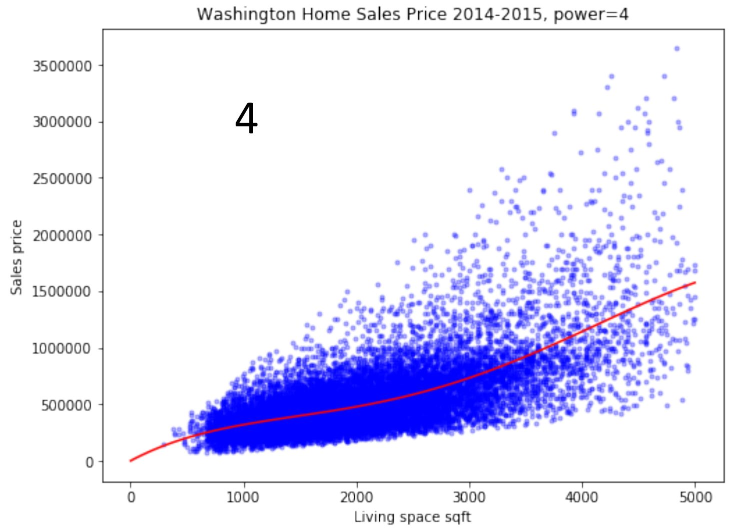
x_1^2

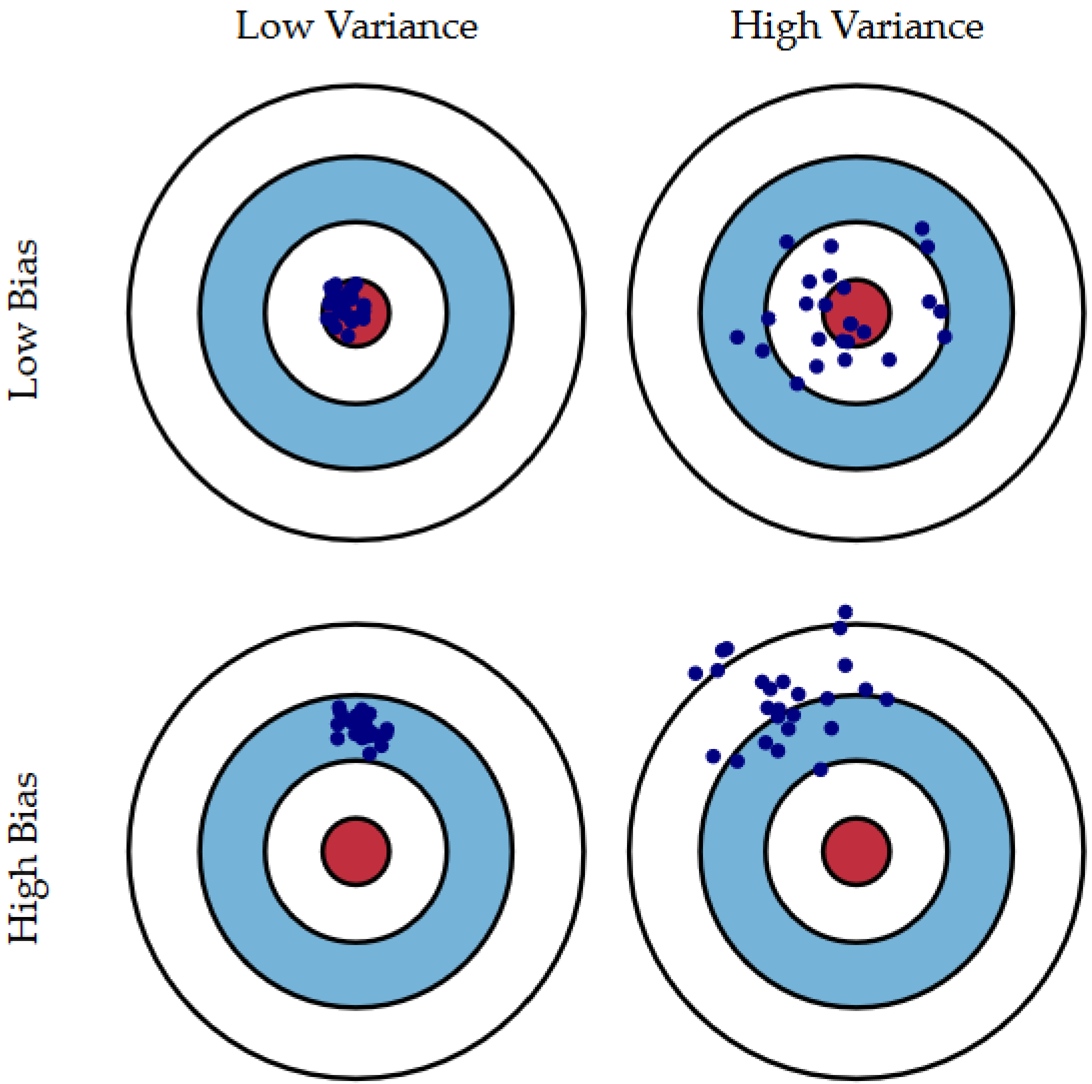
$$\text{MSE} = \frac{1}{2m} \sum_{i}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

$$\hat{y}^{(i)} \in \mathbf{R}$$

Bias-Variance Trade-off





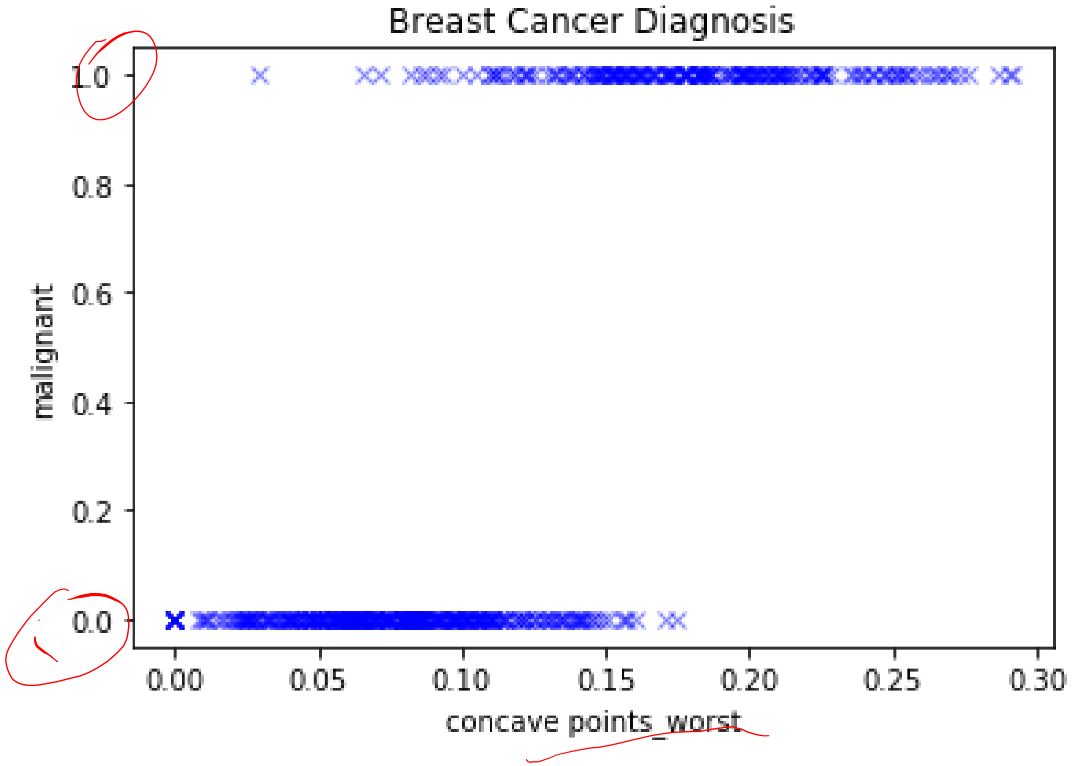


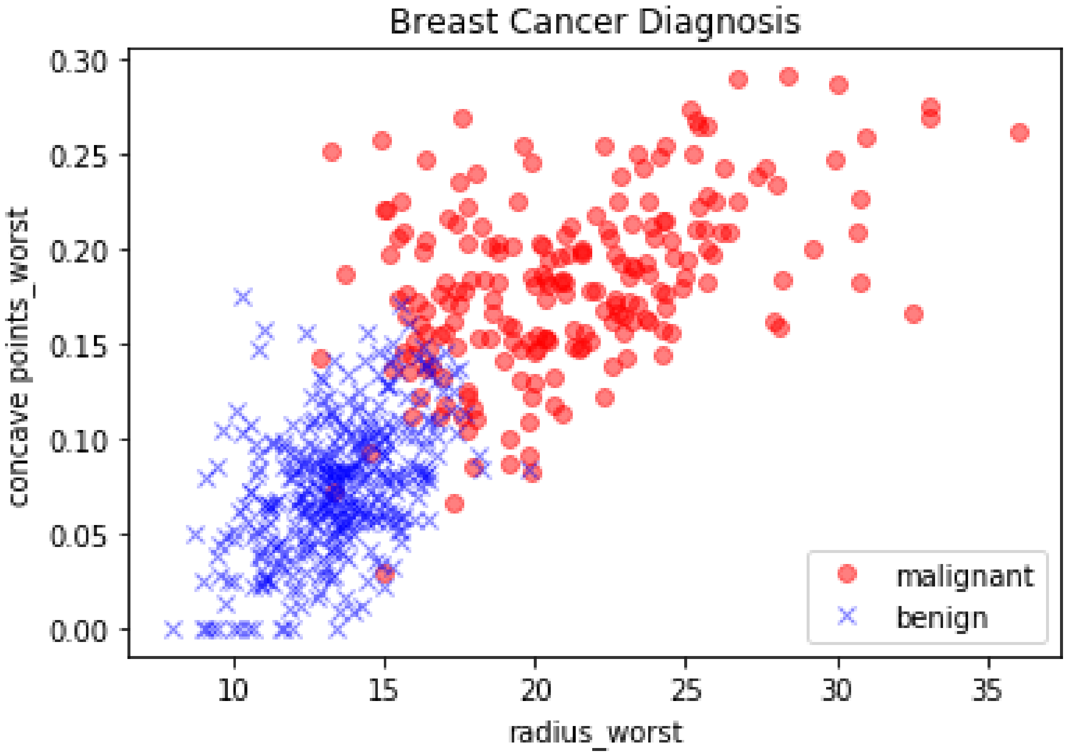
Binary Classification

Yes or No problem

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

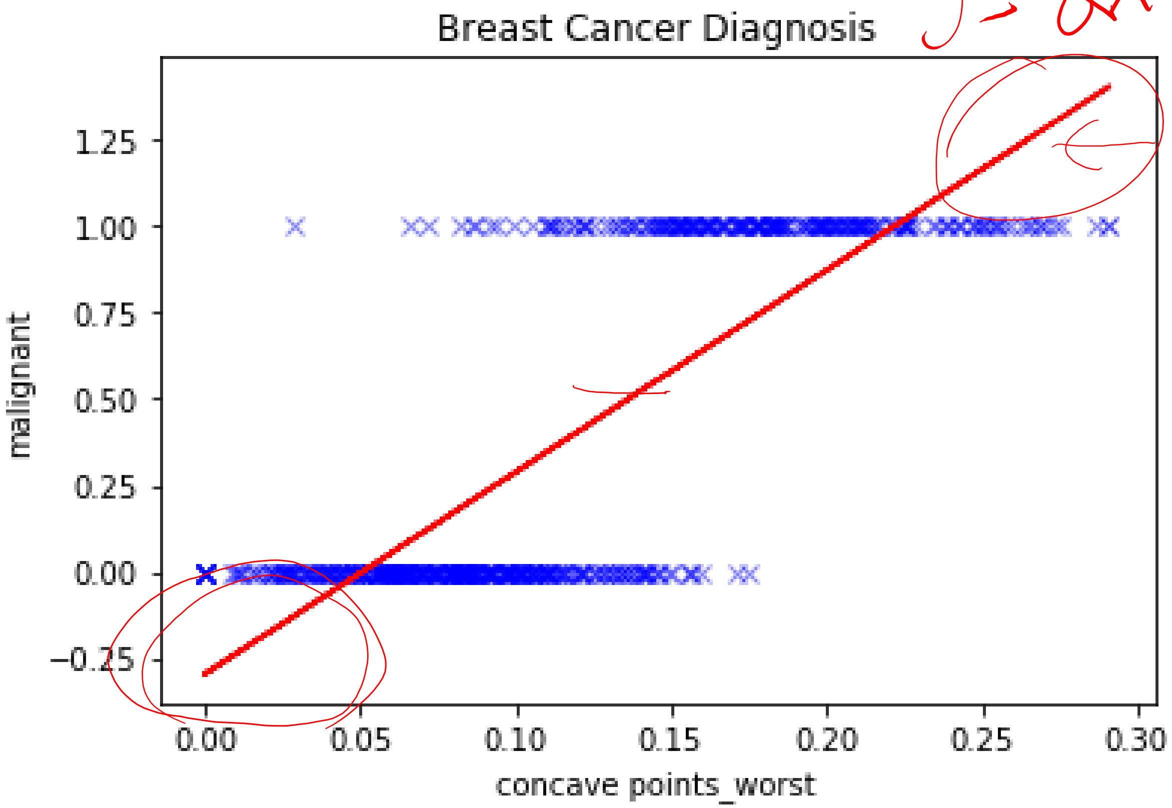
Binary Classification



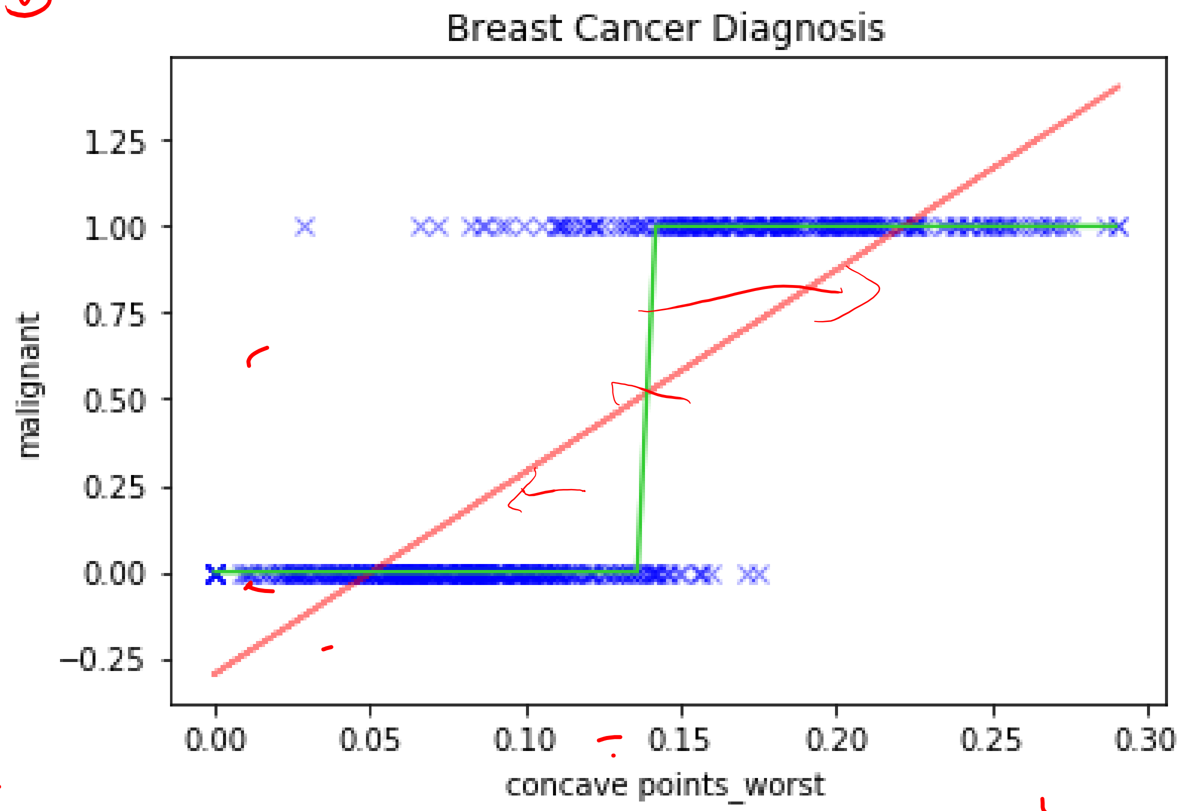


Linear vs Logistic Regression

y = ax + b

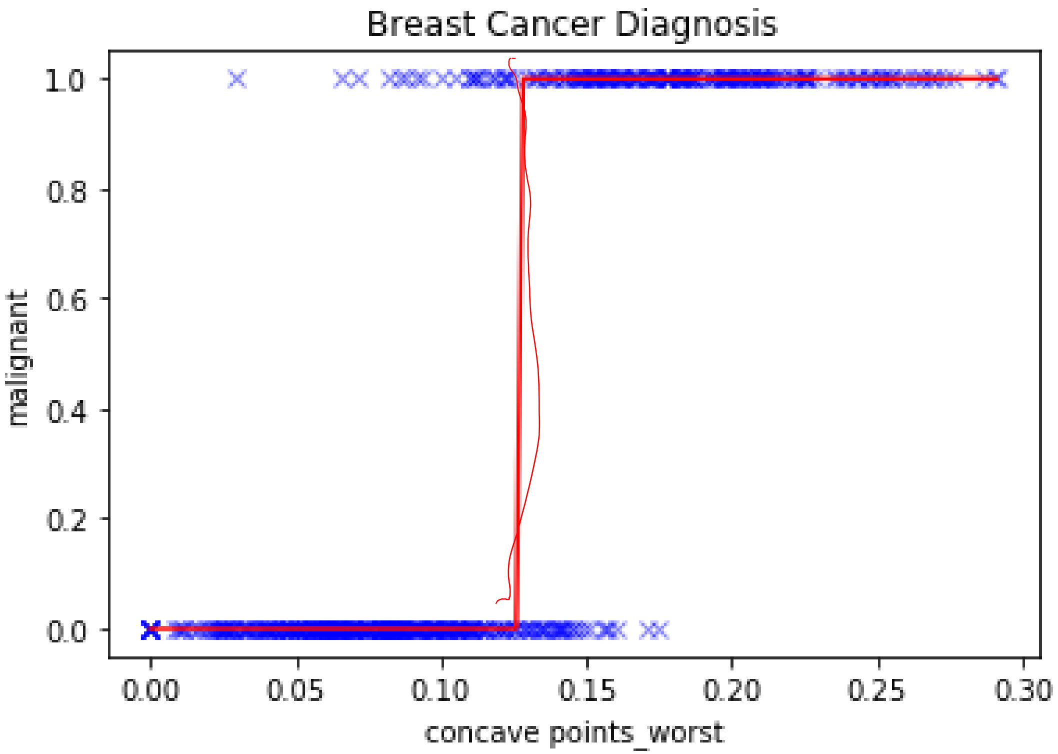


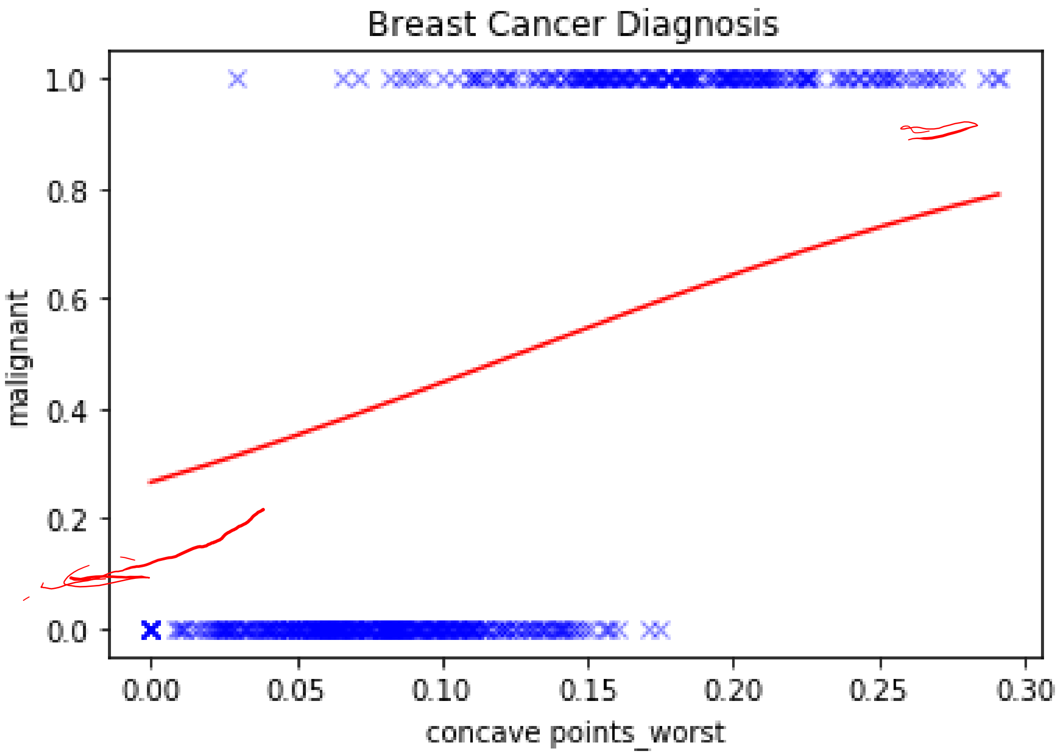
linreg.predict(x)



linreg.predict(x) > 0.5

Linear vs Logistic Regression



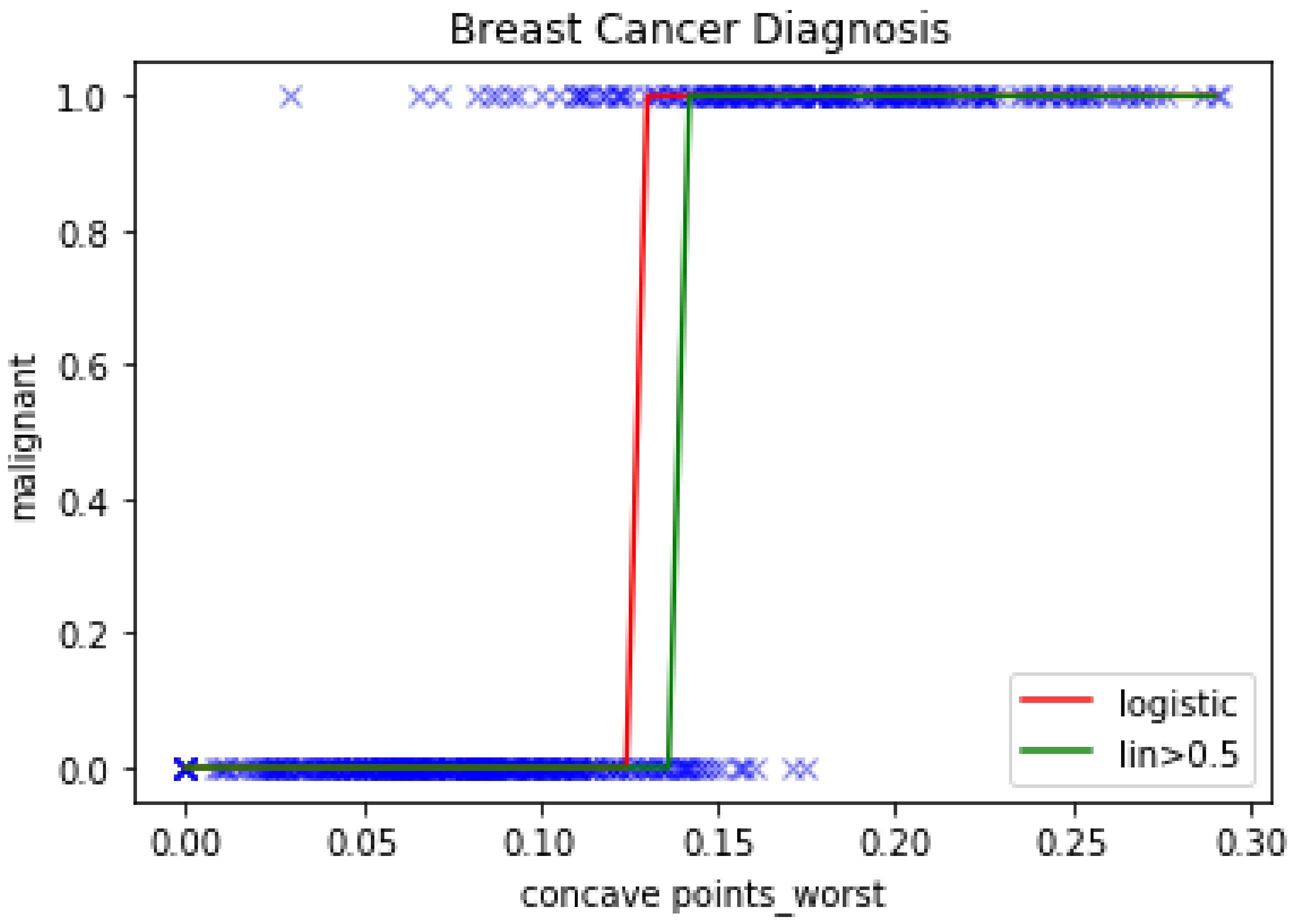


logreg.predict(x)

logreg.predict_proba(x) > 0.5

$\sigma(z) =$

Linear vs Logistic Regression



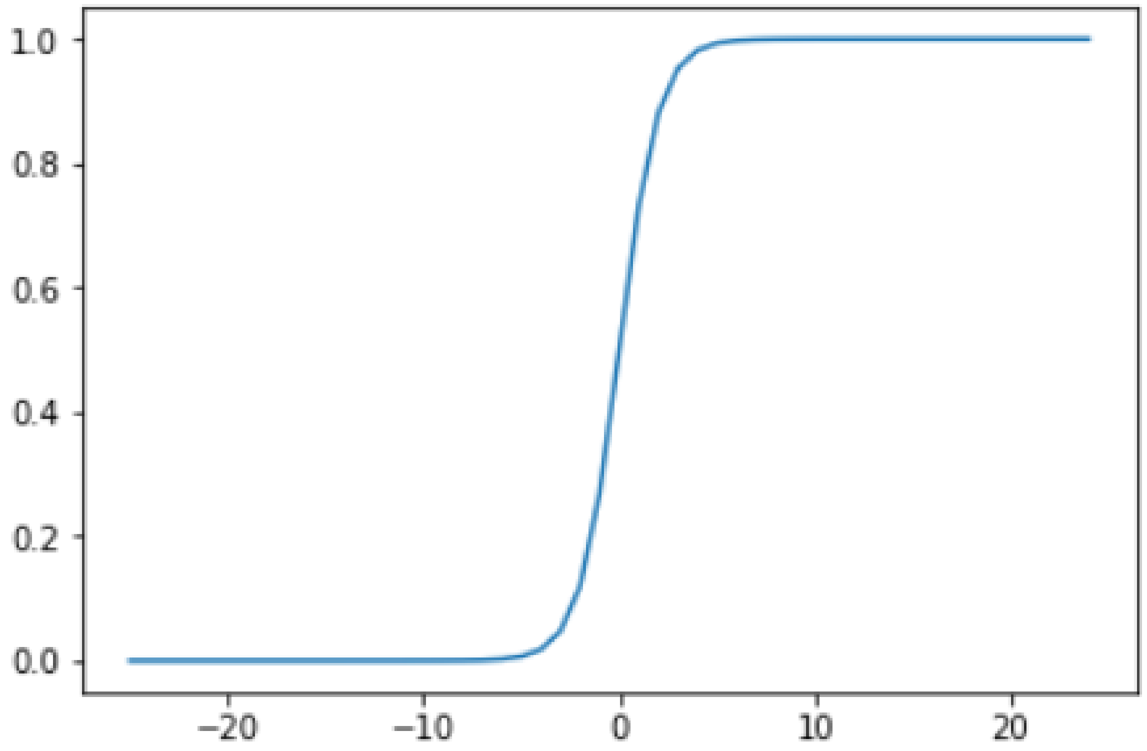
Logistic Function

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

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

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

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

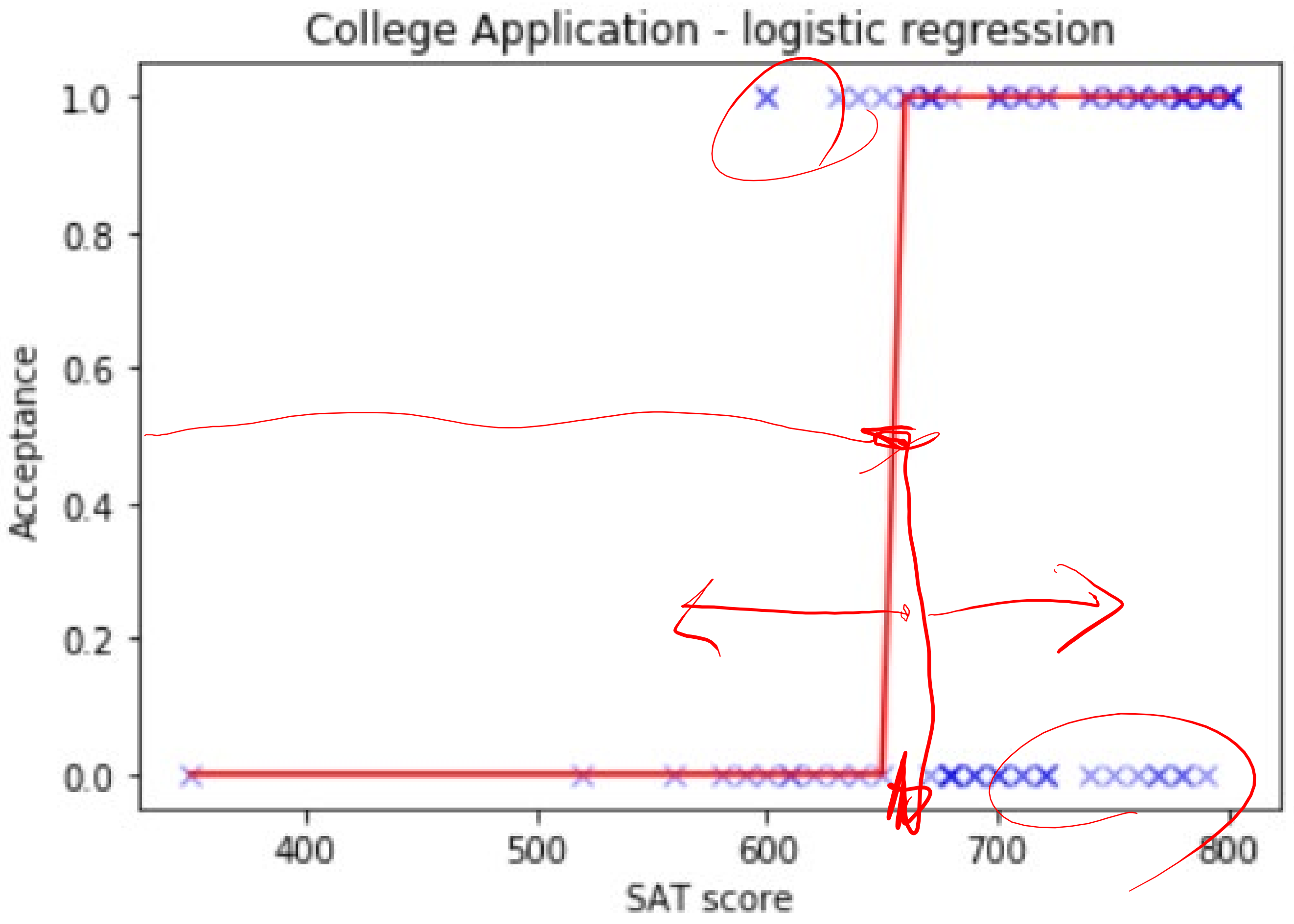


Called "logit" and is related to the decision boundary

Logistic Regression- Univariate

University Acceptance

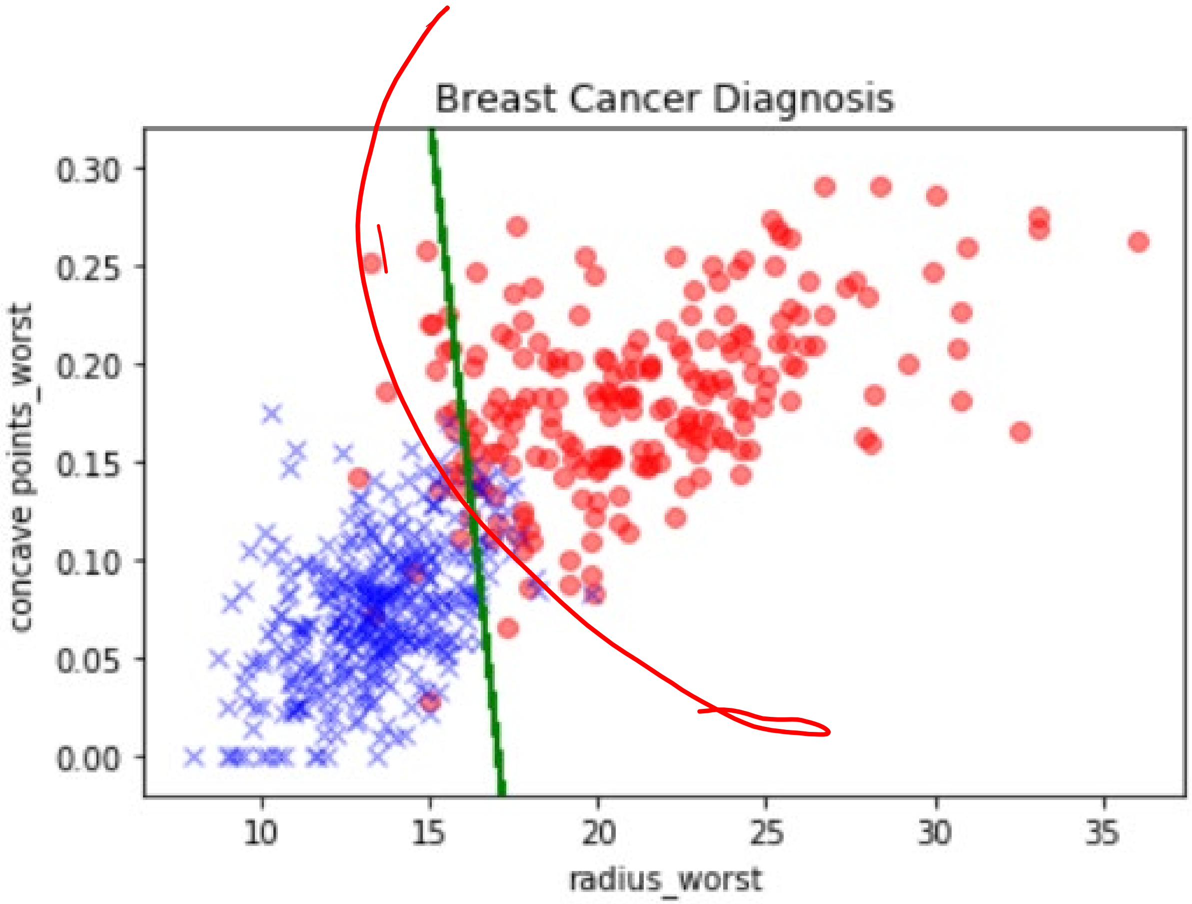
SAT_M	accept
690.0	0.0
710.0	1.0
790.0	1.0
770.0	0.0
770.0	1.0



Logistic Regression- 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 = 0.443 x1 + 2.76 x2 -7.57 = 0

Estimating parameters in logistic regression

Maxmum Likelihood

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

$$= -\sum_i y_i \log P_i + (1-y_i) \log (1-P_i)$$

$\beta_0 \; \beta \ldots$

$\cancel{\operatorname{argmin}_{\beta} \mathcal{L}}$

$\cancel{\mathcal{L} = -\log(ML)}$

Estimating parameters in logistic regression

Cross Entropy

real

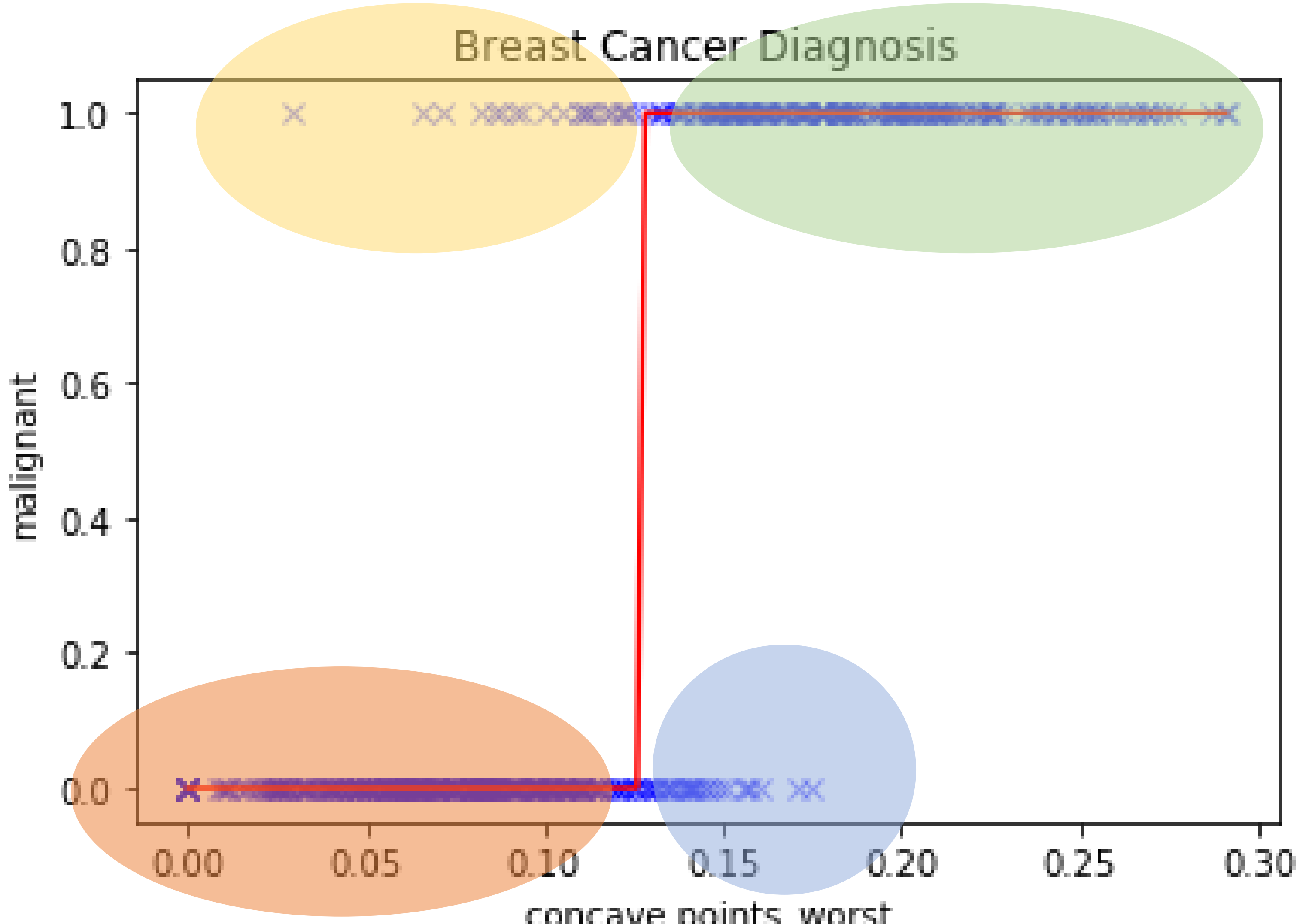
0

1

$$\mathcal{H}(P,Q) = -\sum_i P_i \log(Q_i)$$

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

Interpreting Logistic Regression Result



Yt = 1, Yp = 1
True Positive

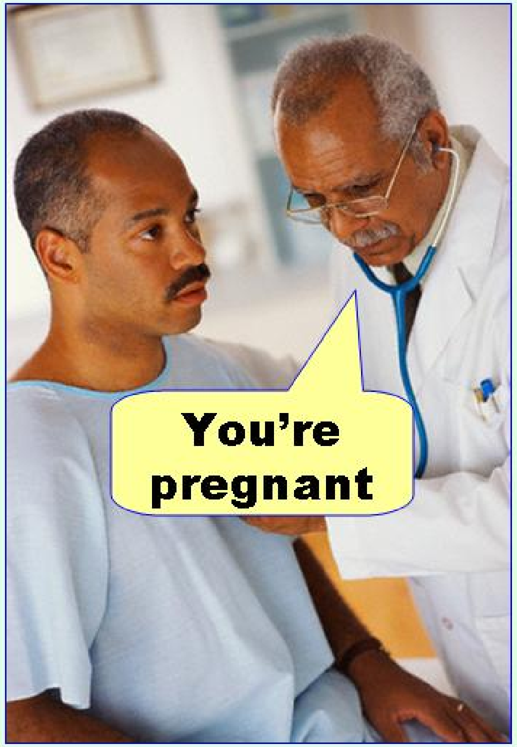
Yt = 0, Yp = 0
True Negative

Yt =0, Yp = 1
False Positive

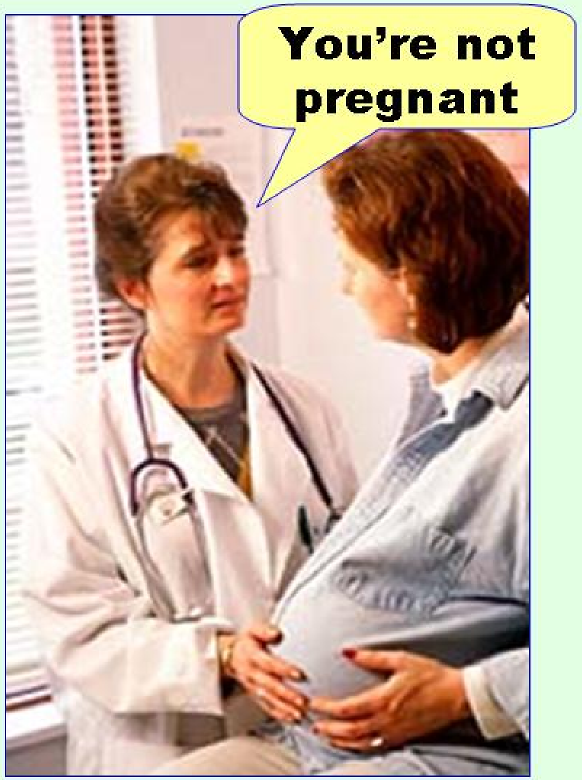
Yt = 1, Yp = 0
False Negative

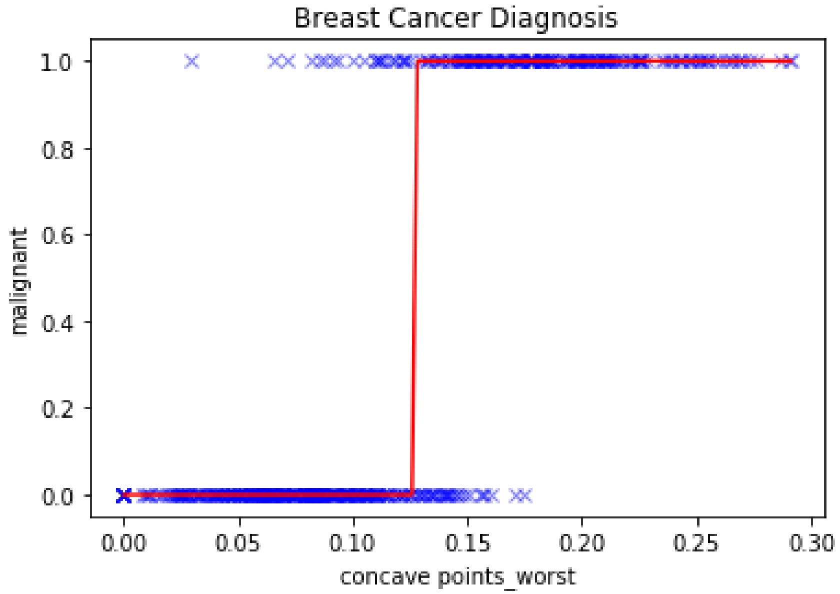
Type I error and Type II error

Type I error
(false positive)



Type II error
(false negative)





Binary Classification Performance Metrics

Confusion Matrix

Yp

Yt

	0	1
0	70	1
1	3	40

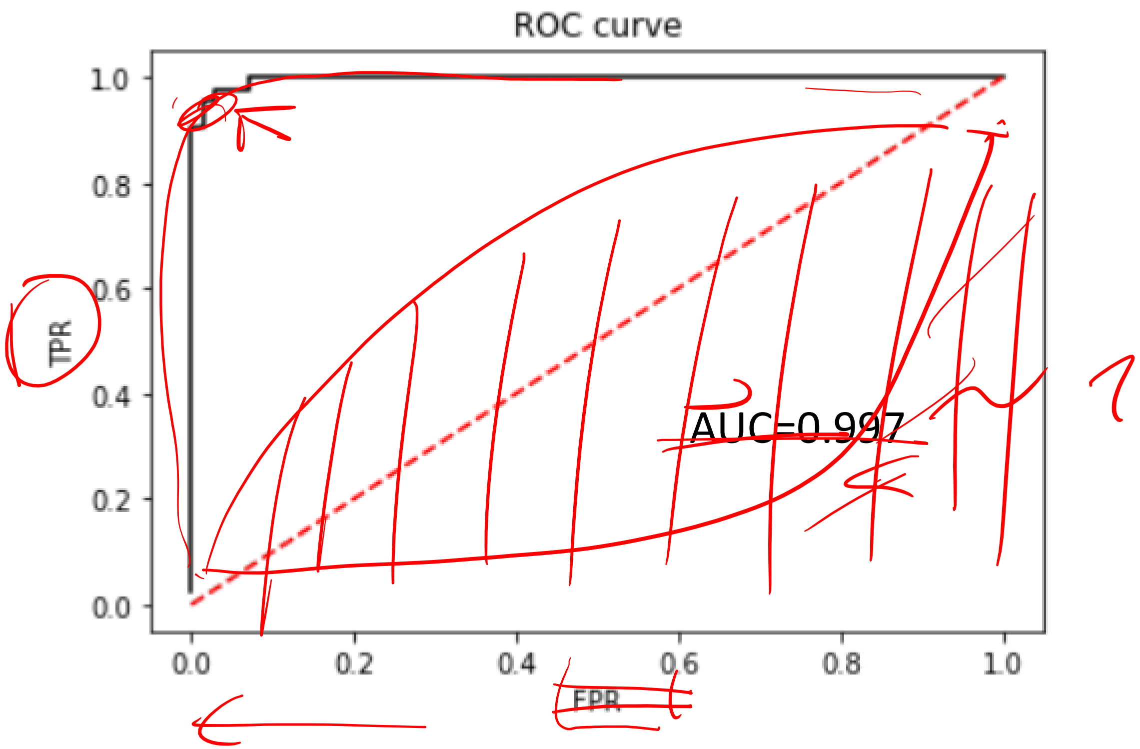
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true, y_pred)
```

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

Performance Metrics- ROC, AUC

Receiver-Operating Characteristics Curve

$\frac{TP}{P}$



Which Performance Metric should I choose?

- Accuracy
- Sensitivity, Recall, TPR
- Specificity, Selectivity, TNR
- Precision, PPV
- False Positive Rate (fall-out)
- False Negative Rate (miss rate)
- F1 score
- AUC
- ~~Confusion matrix~~

999 $y=1$ → 99.9

1 $y=0$

1 2

3

Loss: Why use Cross-Entropy, not Accuracy?

Cross Entropy

$$\mathcal{H}(P,Q) = -\sum_i P_i \log(Q_i) = -\frac{1}{m}\sum_i^m y_i \log \hat{p}_i + (1-y_i) \log (1-\hat{p}_i)$$

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

Scikit-Learn's logistic regression

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='warn', max_iter=100, multi_class='warn', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

```
from sklearn.linear_model import LogisticRegression

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

model.predict(X_test)

model.predict_proba(X_test)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.decision_function

https://github.com/scikit-learn/scikit-learn/blob/1495f6924/sklearn/linear_model/logistic.py

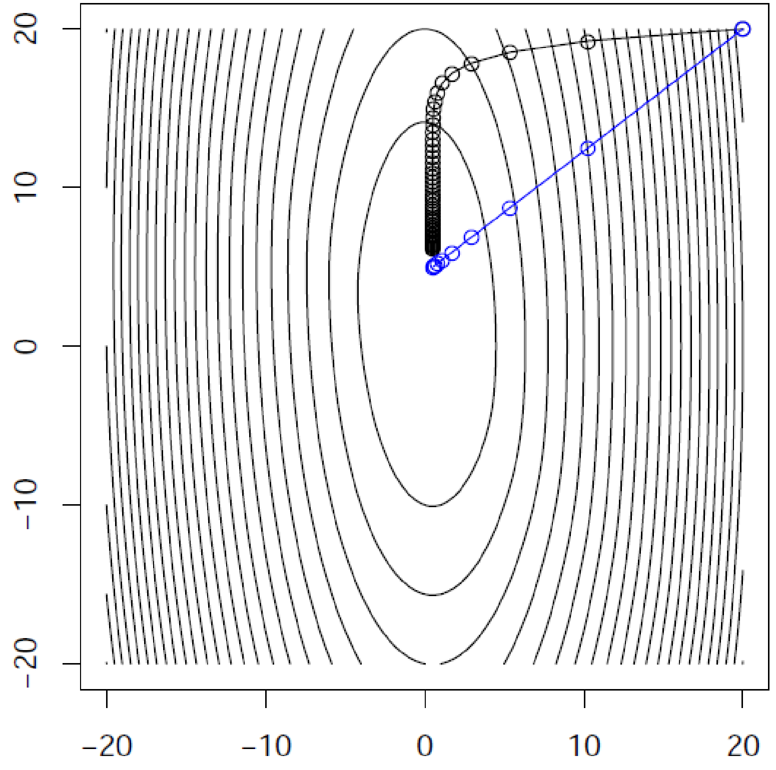
Under the Hood of sklearn's logistic regression

solver : str, {‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, optional (default=’liblinear’).

liblinear (variant of Newton's method)

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$$

$$x^+ = x - (\nabla^2 f(x))^{-1} \nabla f(x)$$



Using sklearn's LogisticRegression

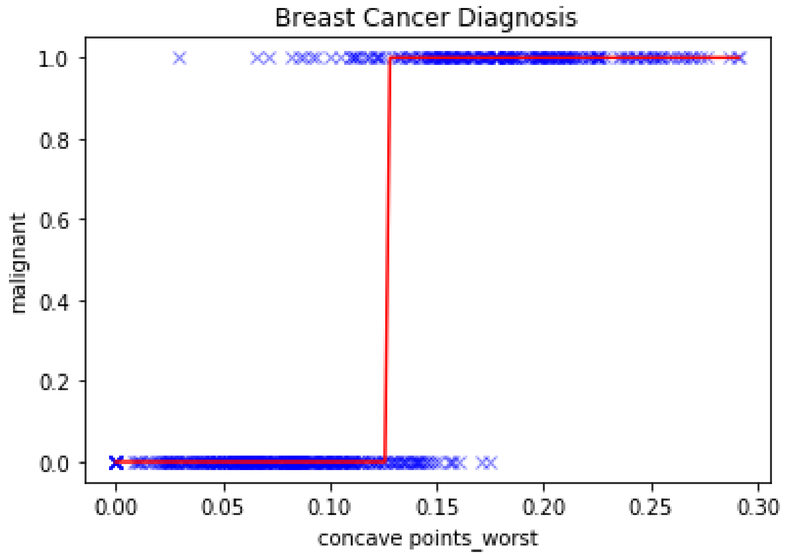
```
from sklearn.linear_model import LogisticRegression

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

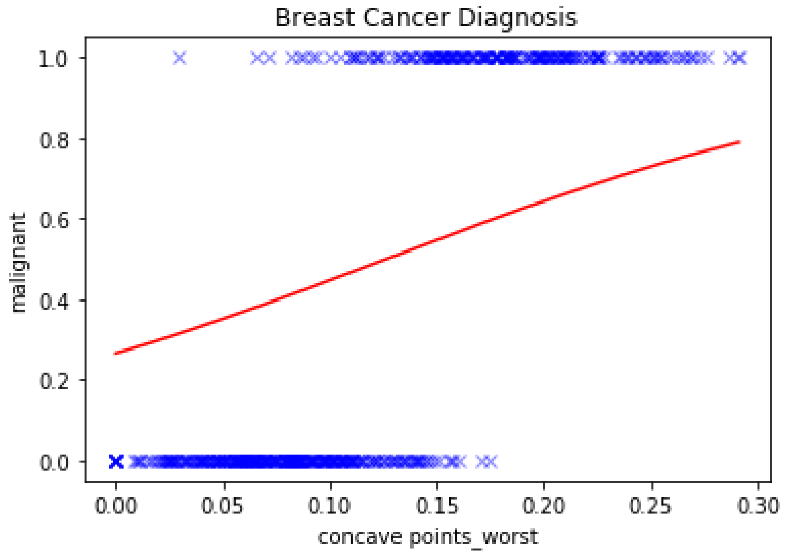
model.coef_

model.intercept_

Yp = model.predict(X_test)

P = model.predict_proba(X_test)





Using sklearn's LogisticRegression

```
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
```

```
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
```

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

	0	1
0	70	1
1	3	40

What about the statistics?

Another 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
==============================================================================
```

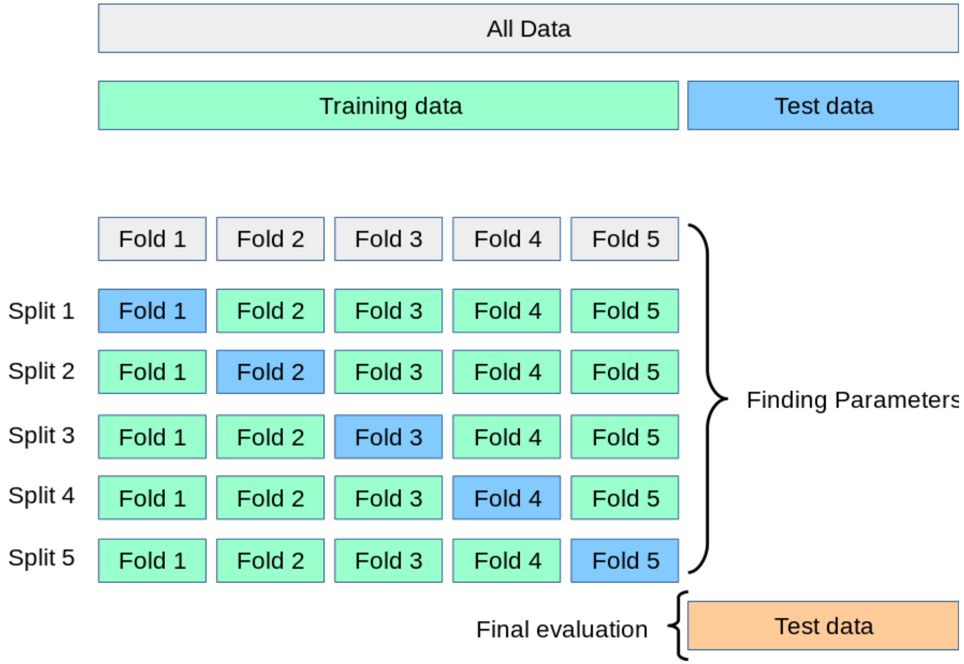
Bootstrap (Resample)

Next Lecture: Ways to train better

Regularization

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='warn', max_iter=100, multi_class='warn', verbose=0, warn_start=False, n_jobs=None, l1_ratio=None*)

Cross-Validation



class
sklearn.linear_model.LogisticRegressionCV