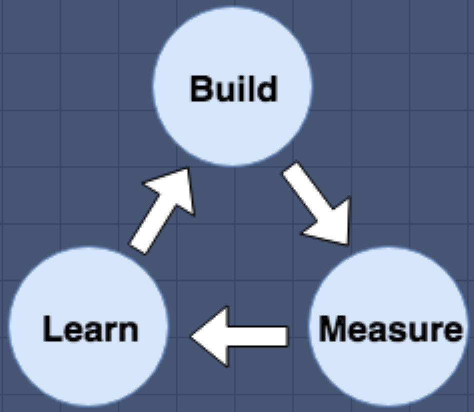
Data Science Camp

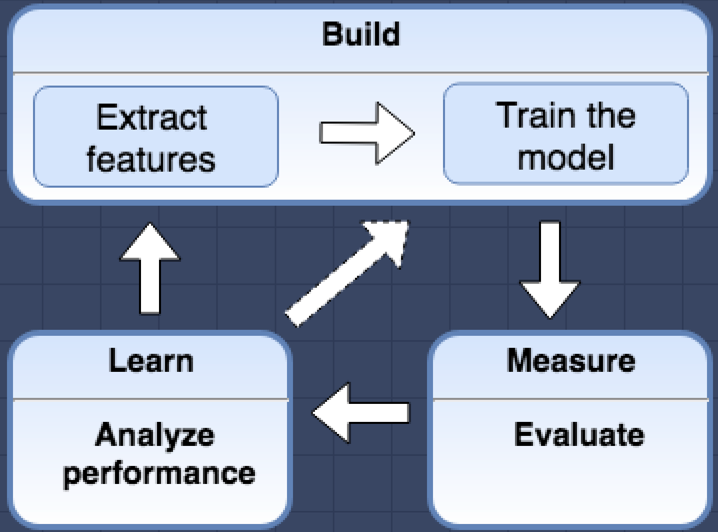
Evaluation

This Lesson

- Evaluation metrics for binary classification
- Evaluation metrics for multi-class classification
- Error analysis
- Regression evaluation metrics

Evaluation In Model Development Cycle





Classification evaluation metrics

Accuracy - is the most popular metric

Business may induce different metrics e.g.:

- Users satisfaction (web search)
- Revenue (e-commerce)
- Patient survival rates (medical)

Some domains may require shifted threshold of classification e.g. check for cancer requires high accuracy on positive (cover all for sure) though with low accuracy of negative (some false negative prediction)

Imbalanced Classes

E.g. Fraudulent transaction detection

Out of 1,000 random;uy selected:

- 1 is fraudulent (relevant)
- 999 are normal (irrelevant)

Dummy classifier that always predicts the majority class

Accuracy: 99.9%

Accuracy is not good measure for this case

Dummy Classifier

- Completely ignores input data (X)
- Use it as sanity check of your classifier performance
- Consider it as null-metric (baseline)
- Don’t use for real problem

```
from sklearn.dummy import  DummyClassifier
clf= DummyClassifier(strategy='most_frequent').fit(X_train, y_train)
clf.predict(X_test)
print("train accuracy= {:.3%}".format(clf.score (X_train, y_train)))
print("test accuracy= {:.3%}".format(clf.score (X_test, y_test)))
```

*

Dummy Classifier Parameters

Parameters:

- strategy : str, default="stratified"
  - "most_frequent": always predicts the most frequent label in the training set.
  - "stratified": generates predictions by respecting the training set's class distribution.
  - "uniform": generates predictions uniformly at random.
  - "constant": always predicts a constant label that is provided by the user.

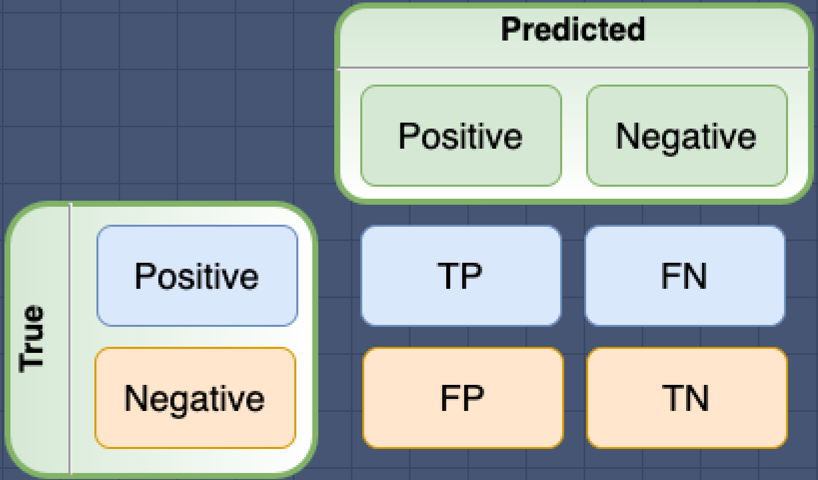
    (useful for metrics that evaluate a non-majority class)
- random_state : int, default=None
- constant : int or str or array of shape = [n_outputs]

Null-Metric Cases

Reasons of your classifier accuracy is close to dummy classifier accuracy:

- wrong/missed relevant features
- wrong parameters (e.g. kernel)
- Imbalanced data

If imbalanced data use another evaluation metric



Confusion Matrix (матриця помилок)

Confusion Matrix For Dummy Classifier

Dummy Stratified		Predicted: Positive	Predicted: Negative
True	Positive	TP = 5	FN = 38
	Negative	FP = 37	TN = 370

Let's have training set that consists of 1347 samples:

1208 negative and 139 positive.

Thus stratified dummy classifier predicts with distribution ratio = 0.115. Prediction may vary on random state. Let's consider one of them.

Accuracy

Dummy Stratified		Predicted: Positive	Predicted: Negative
True	Positive	TP = 5	FN = 38
	Negative	FP = 37	TN = 370

What is the share of correct predicted from total ?

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{375}{450} = 0.83$$

Classification Error

Dummy Stratified		Predicted: Positive	Predicted: Negative
True	Positive	TP = 5	FN = 38
True	Negative	FP = 37	TN = 370

What is the share of incorrect predicted from total ?

$$classification\,error = \frac{FP + FN}{TP + TN + FP + FN} = \frac{75}{450} = 0.17$$

Precision

Dummy Stratified		Predicted Positive	Predicted Negative
True	Positive	TP = 5	FN = 38
	Negative	FP = 37	TN = 370

What is the share of
correct predicted positive
from all predicted positive ?

$$precision = \frac{TP}{TP + FP} = \frac{5}{42} = 0.119$$

Recall

Dummy Stratified		Predicted: Positive	Predicted: Negative
True	Positive	TP = 5	FN = 38
True	Negative	FP = 37	TN = 370

What is the share of correct predicted positive from all true positive?

= True Positive Rate (TPR)

= Sensitivity

= Probability Of Detection

$$recall = \frac{TP}{TP + FN} = \frac{5}{43} = 0.116$$

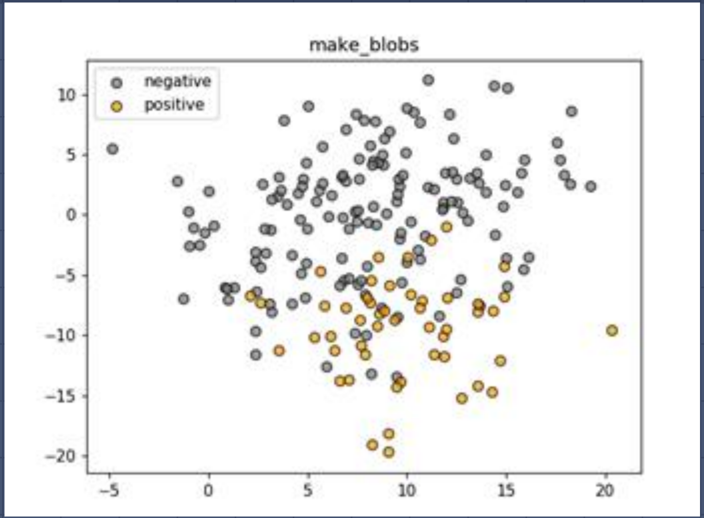
False Positive Rate

Dummy Stratified		Predicted Positive	Predicted Negative
True	Positive	TP = 5	FN = 38
True	Negative	FP = 37	TN = 370

= Specificity

$$falsepositiverate = \frac{FP}{FP + TN} = \frac{37}{407} = 0.09$$

Logistic Regression for Synthetic Data Set



Logistic Regression

Probability Of Prediction

index	true_value	predicted	probability of 0	probability of 1
46	True	True	0.215176	0.784824
47	True	False	0.554316	0.445684
44	False	False	0.982009	0.017991
22	True	True	0.042861	0.957139
10	True	True	0.077809	0.922191
45	False	False	0.948227	0.051773
31	False	False	0.956127	0.043873
42	True	True	0.213881	0.786119
3	False	False	0.532526	0.467474
33	True	True	0.252123	0.747877

clf.predict_proba(X_test)

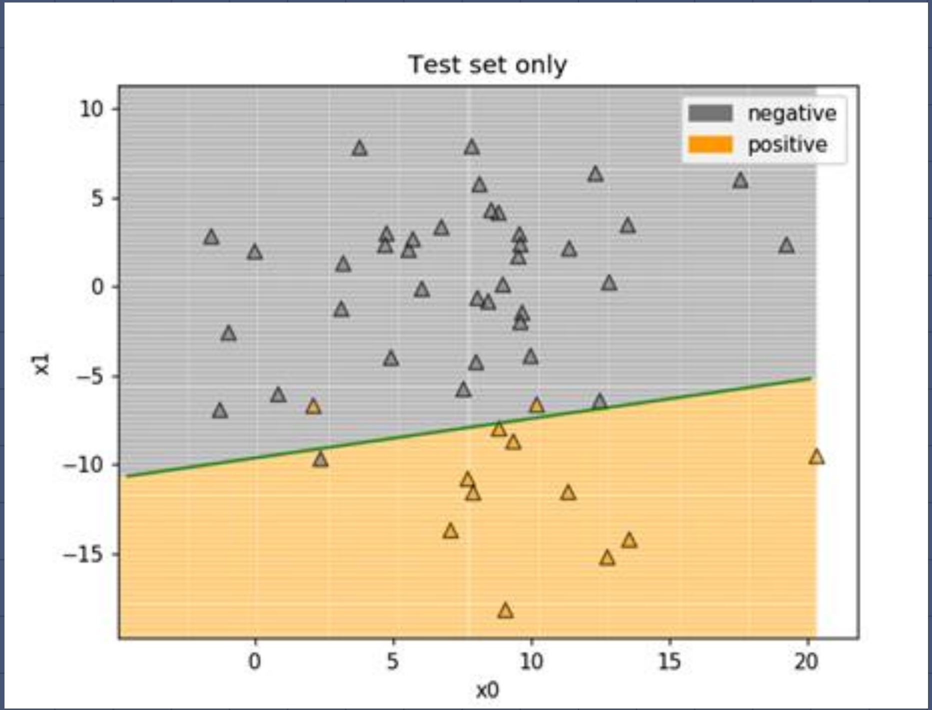
Threshold = 0.5

Decision Function

index	true_value	predicted	score
37	True	False	0.334655
23	False	False	-3.080418
44	False	False	-3.999724
42	True	True	1.301689
47	True	False	-0.218125
20	False	False	-2.039472
3	False	False	-0.130288
30	False	False	-2.122499
7	False	False	-4.643650
6	False	False	-4.890379

```
y_score = clf.decision_function(X_test)
```

Performance On Test Set

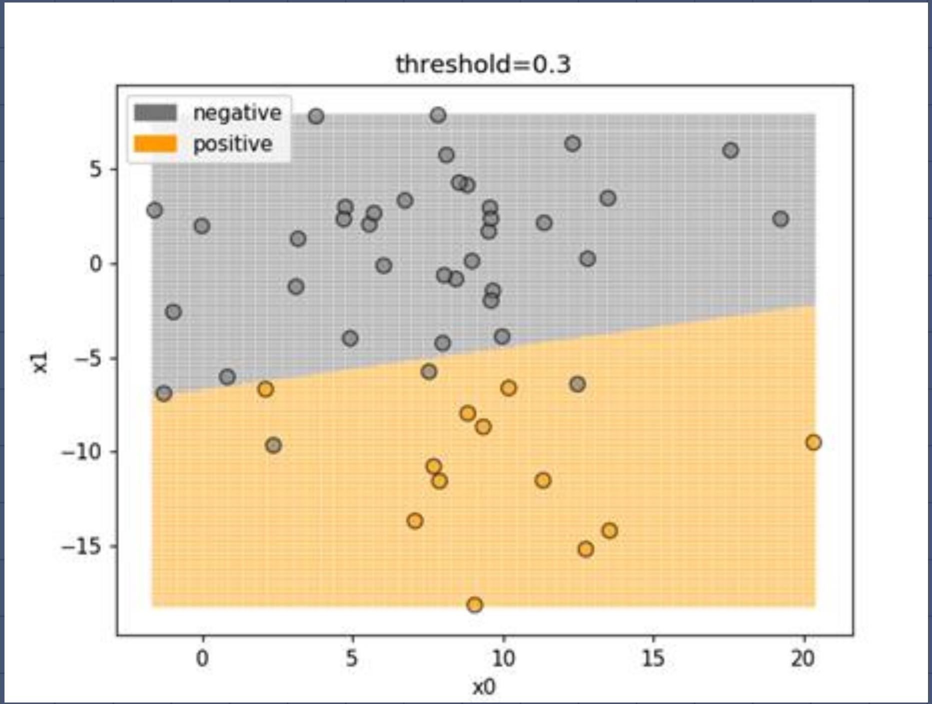


	Predicted Negative	Predicted Positive
True Negative	TN = 37	FP = 1
True Positive	FN = 2	TP = 10

Recall = 0.91

Precision = 0.83

Recall Oriented Classifier



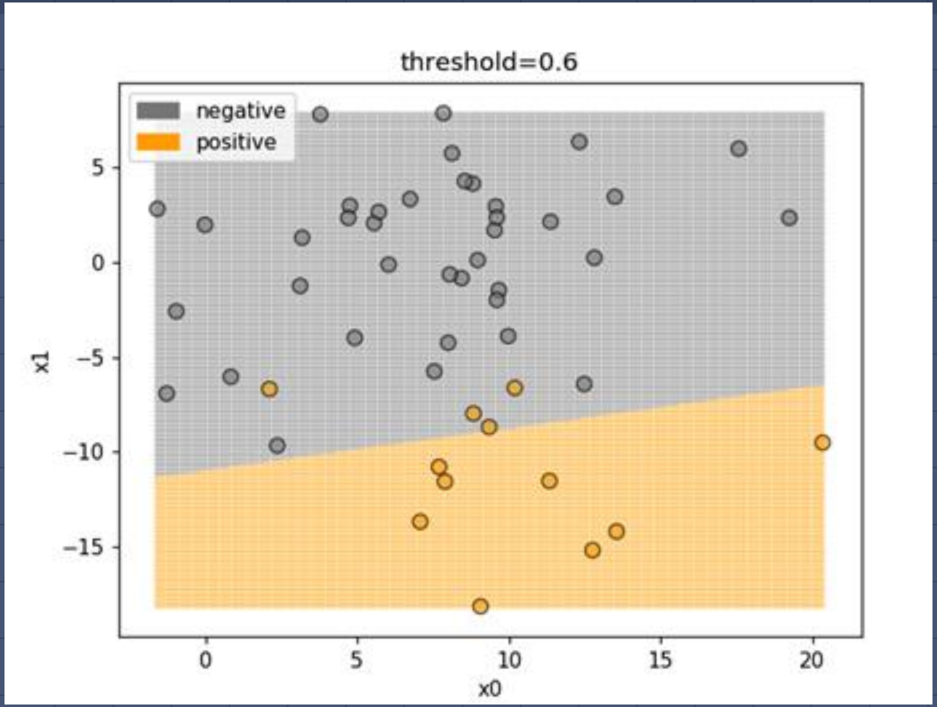
	Predicted Negative	Predicted Positive
True Negative	TN = 35	FP = 3
True Positive	FN = 0	TP = 12

Recall = 1.00

Precision = 0.80

Avoid False Negative (e.g. make sure every positive diagnose is in account)

Precision Focused Classifier



	Predicted Negative	Predicted Positive
True Negative	TN = 38	FP = 0
True Positive	FN = 4	TP = 8

Recall = 0.67

Precision = 1.00

Avoid False Positive (e.g. recommend only matched movies and no partially matched)

Aggregated Score

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

$$F_\beta = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} = \frac{(1+\beta^2) \cdot TP}{(1+\beta^2) \cdot TP + \beta \cdot FN + FP}$$

β* - *Determines the weight of recall in the combined score.

β > 1, more weight on recall

β < 1, more weight on precision

β = 1, same weight, basically same as F1 score

Sklearn Metrics

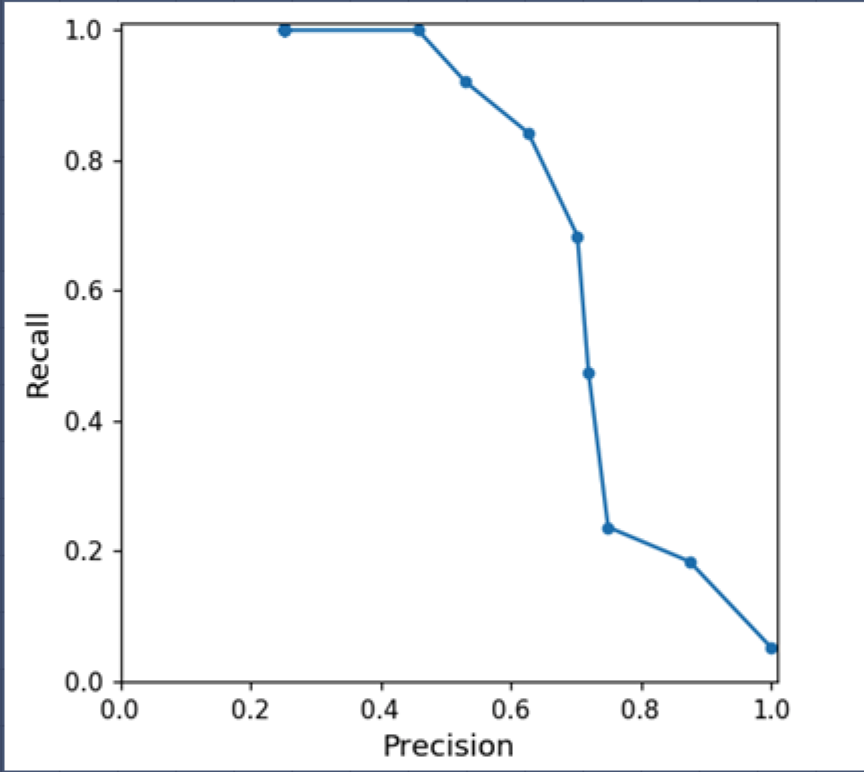
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
y_predicted = clf.predict(X_test)
print ('accuracy = {:.2}'.format(accuracy_score(y_test, y_predicted)))
print ('recall = {:.2}'.format(recall_score(y_test, y_predicted)))
print ('precision = {:.2}'.format(precision_score(y_test, y_predicted)))
print ('f1_score  = {:.2}'.format(f1_score(y_test, y_predicted)))
```

Out:

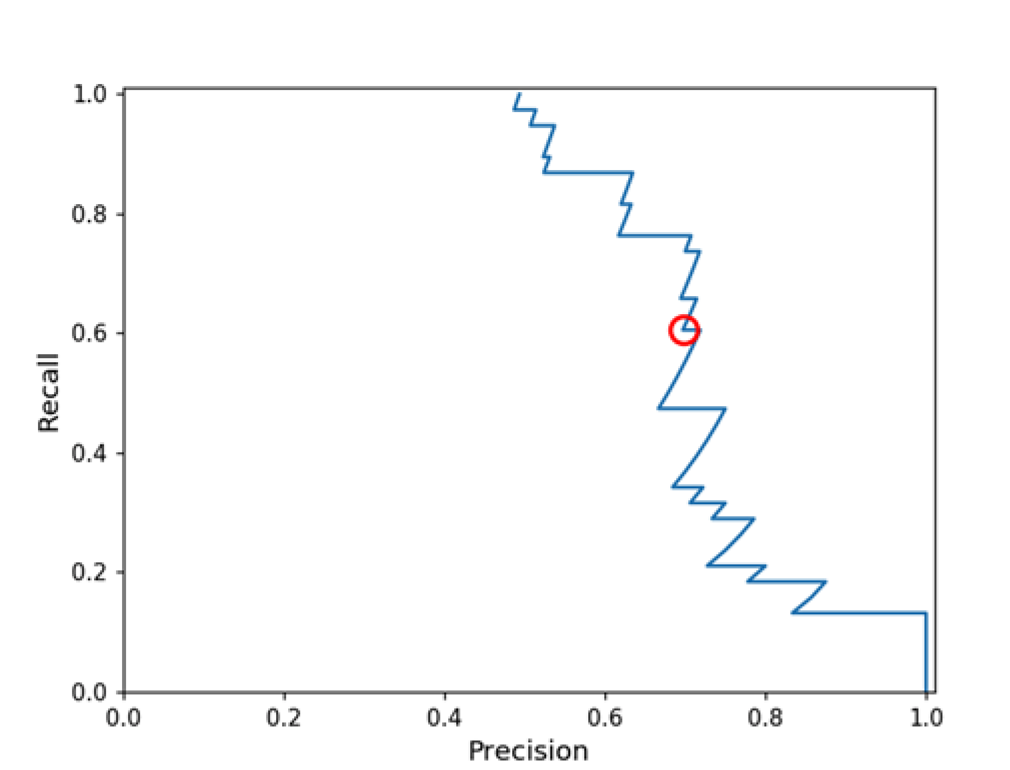
```
accuracy = 0.94
recall = 0.83
precision = 0.91
f1_score  = 0.87
```

Precision-Recall Curve

threshold	recall	precision
0.017282	1.000000	0.253333
0.104312	1.000000	0.457831
0.225905	0.921053	0.530303
0.347498	0.842105	0.627451
0.469091	0.684211	0.702703
0.590685	0.473684	0.720000
0.712278	0.236842	0.750000
0.833871	0.184211	0.875000
0.955465	0.052632	1.000000



Precision-Recall Curve



Train set (150 samples) of synthetic data set (make_blobs) after training Logistic Regression classifier

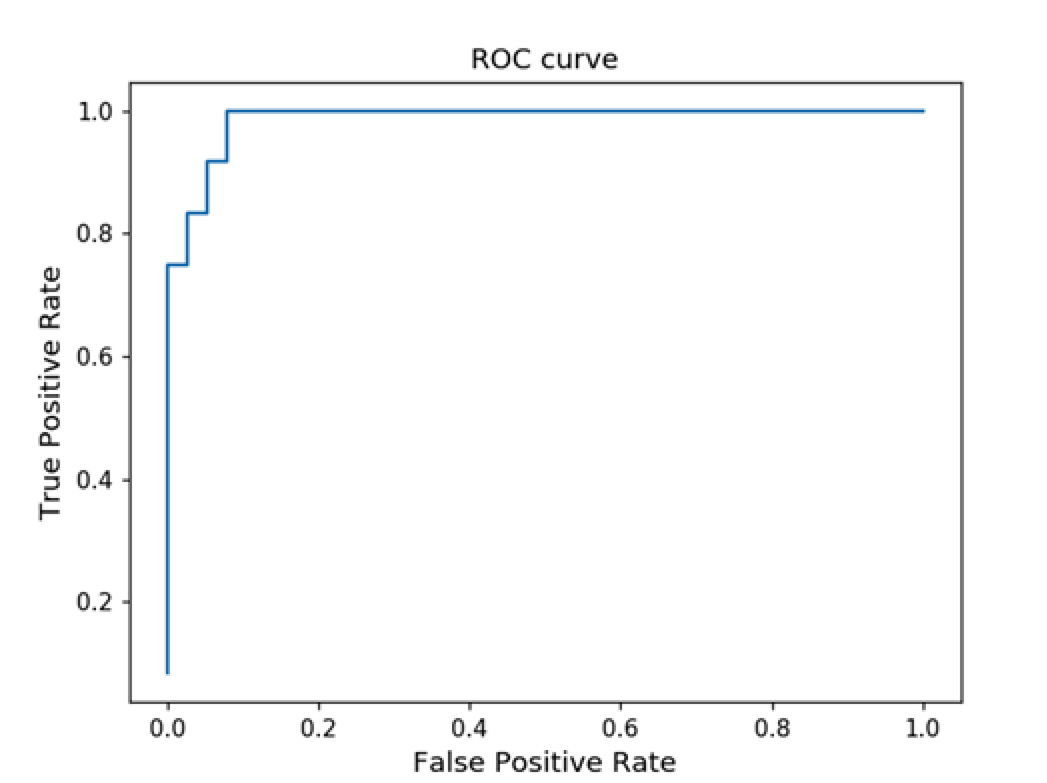
from sklearn.metrics import precision_recall_curve

y_score = clf.decision_function(X_test)

precision, recall, thresholds = precision_recall_curve(y_test, y_score)

* Precision Recall Curve

Receiver Operating Characteristic (ROC) Curve



Test set (50 samples) of synthetic data set (make_blobs) after training Logistic Regression classifier

from sklearn.metrics import roc_curve

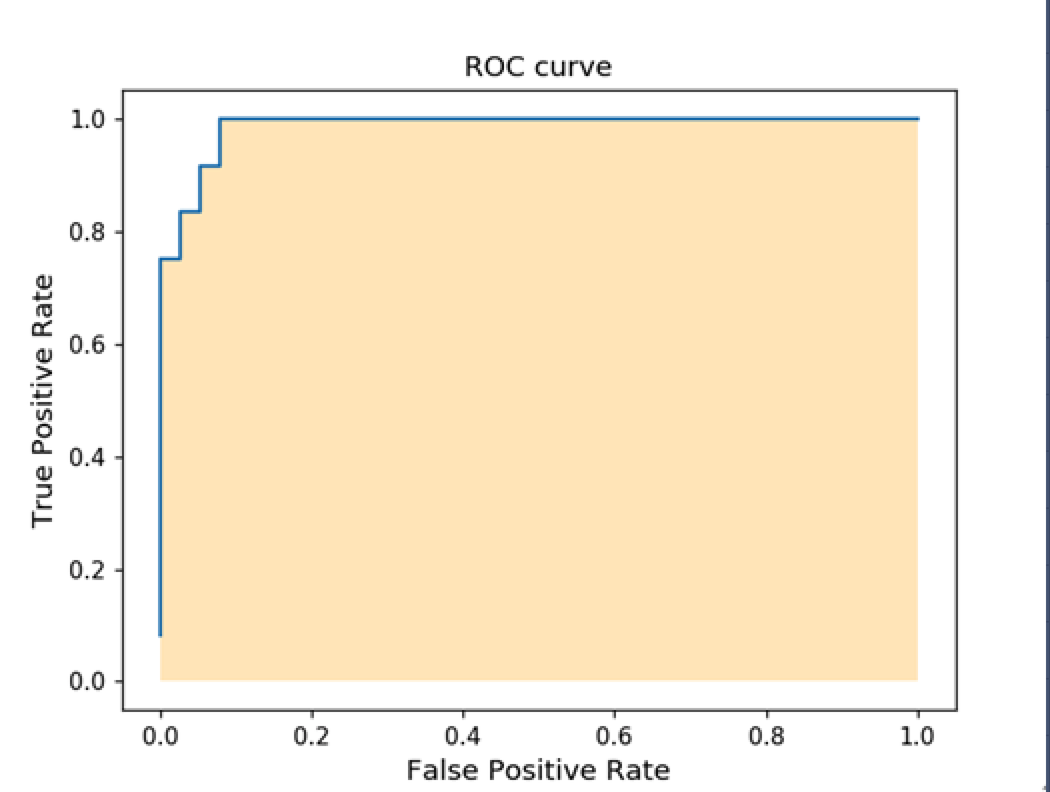
precision, recall, thresholds = roc_curve(y_test, y_score)

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

*ROC curve

AUC Score



Test set (50 samples) of synthetic data set (make_blobs) after training Logistic Regression classifier

from sklearn.metrics import auc

precision, recall, thresholds = roc_curve(y_test, y_score)

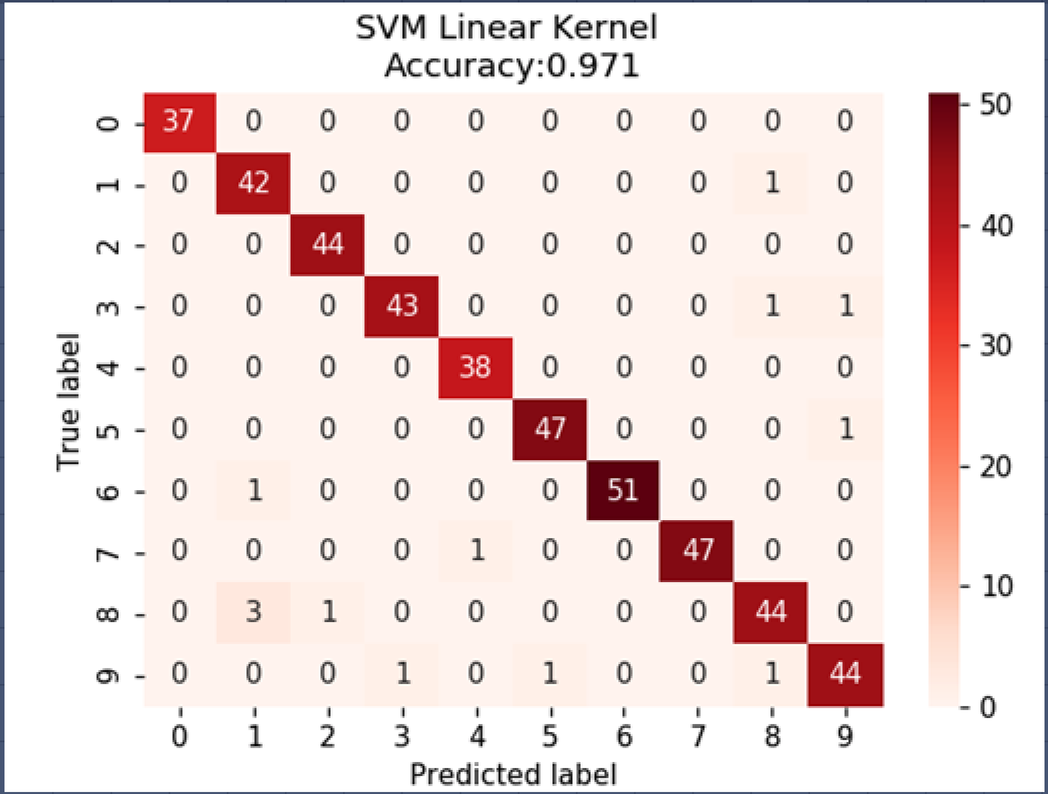
roc_auc = auc(fpr, tpr)

Out:

roc_auc = 0.99

*AUC score

Multi-Class Confusion Matrix



from sklearn.datasets import

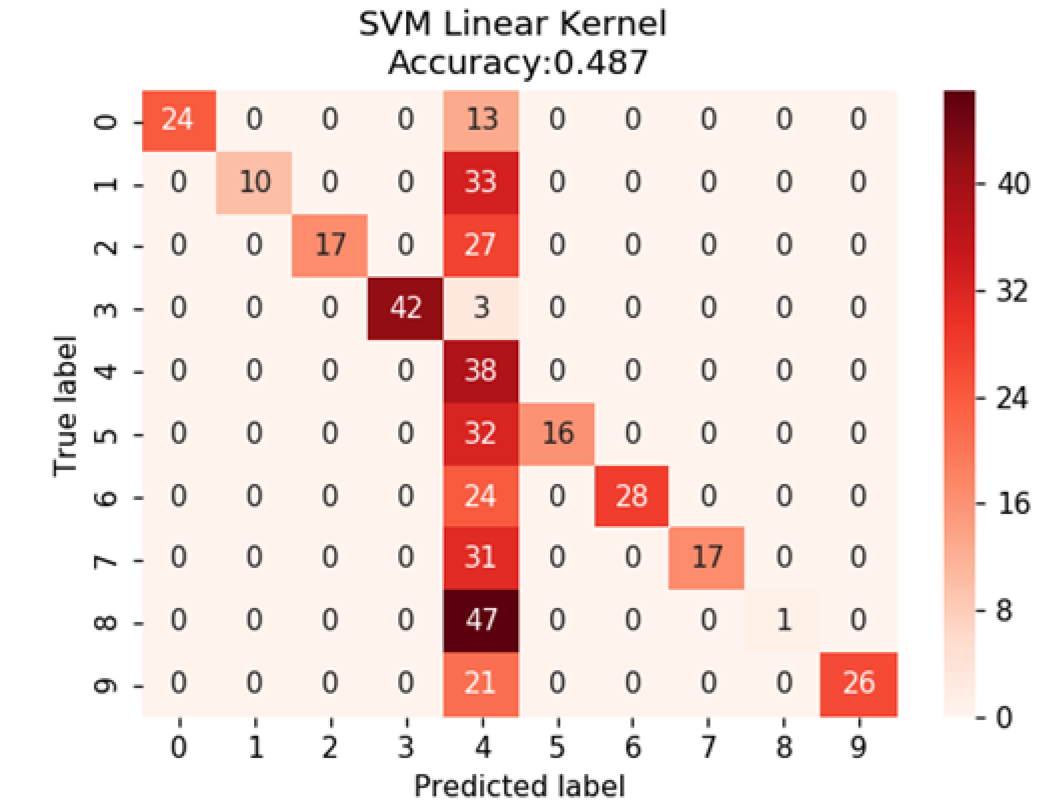
load_digits

from sklearn.svm import SVC

Linear kernel

import seaborn as sns

Multi-Class Confusion Matrix



from sklearn.datasets import

load_digits

from sklearn.svm import SVC

RBF kernel

import seaborn as sns

*Multi-class confusion matrix

Macro-Average Precision

Micro vs Macro Average

Class	Predicted Class	Correct?
orange	lemon	0
orange	lemon	0
orange	apple	0
orange	orange	1
orange	apple	0
lemon	lemon	1
lemon	apple	0
apple	apple	1
apple	apple	1

Macro-average:

- Each class has equal weight.

1. Compute metric within each class
2. Average resulting metrics across classes

Class	Precision
orange	1/5 = 0.20
lemon	1/2 = 0.50
apple	2/2 = 1.00

Macro-average precision:
(0.20 + 0.50 + 1.00) / 3 = **0.57**

Micro-Average Precision

Micro vs Macro Average

Class	Predicted Class	Correct?
orange	lemon	0
orange	lemon	0
orange	apple	0
orange	orange	1
orange	apple	0
lemon	lemon	1
lemon	apple	0
apple	apple	1
apple	apple	1

Micro-average:
- Each instance has equal weight.
- Largest classes have most influence

1. Aggregrate outcomes across all classes
2. Compute metric with aggregate outcomes

Micro-average precision:
4 / 9 = **0.44**

Macro vs Micro Average Precision

Macro-Average vs Micro-Average

- If the classes have about the same number of instances, macro- and micro-average will be about the same.
- If some classes are much larger (more instances) than others, and you want to:
  - Weight your metric toward the largest ones, use micro-averaging.
  - Weight your metric toward the smallest ones, use macro-averaging.
- If the micro-average is much lower than the macro-average then examine the larger classes for poor metric performance.
- If the macro-average is much lower than the micro-average then examine the smaller classes for poor metric performance.

*Average

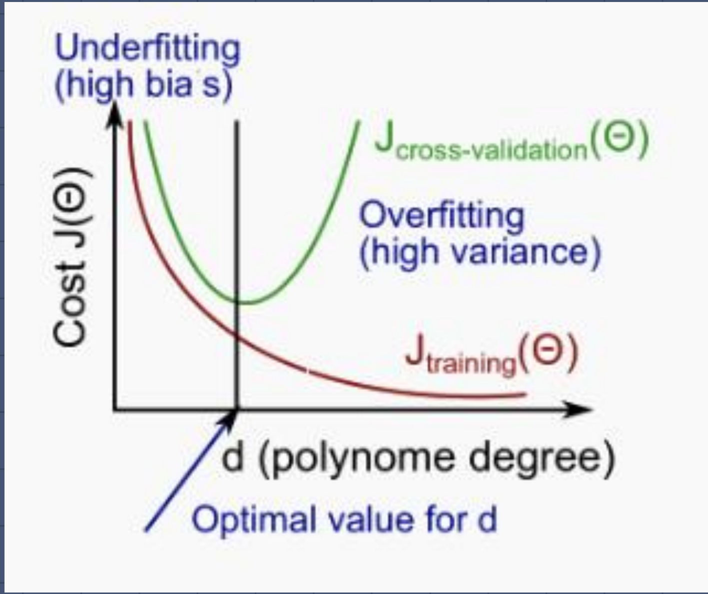
Train/Validation/Test Sets

	outlook	humidity	wind	play
Train	sunny	high	weak	no
	sunny	high	strong	no
	overcast	high	weak	yes
	rainy	high	weak	yes
	rainy	normal	weak	yes
	rainy	normal	strong	no
	overcast	normal	strong	yes
	sunny	high	weak	no
	sunny	normal	weak	yes
	rainy	normal	weak	yes
Validation	sunny	normal	strong	yes
	overcast	high	strong	yes
Test	overcast	normal	weak	yes
	rainy	high	strong	no

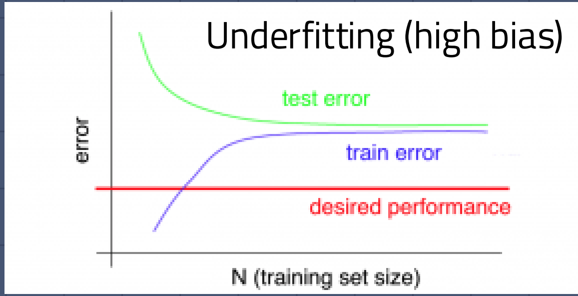
Using test set to tune the model parameters (e.g. degree of polynomial features) tends to provide the most optimistic evaluation

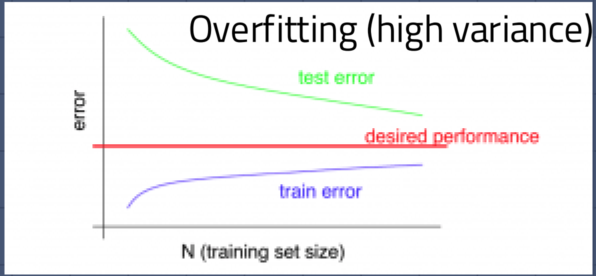
Error Analysis : Bias vs Variance

Comparing classification error of Train and Validation sets depending on parameter value

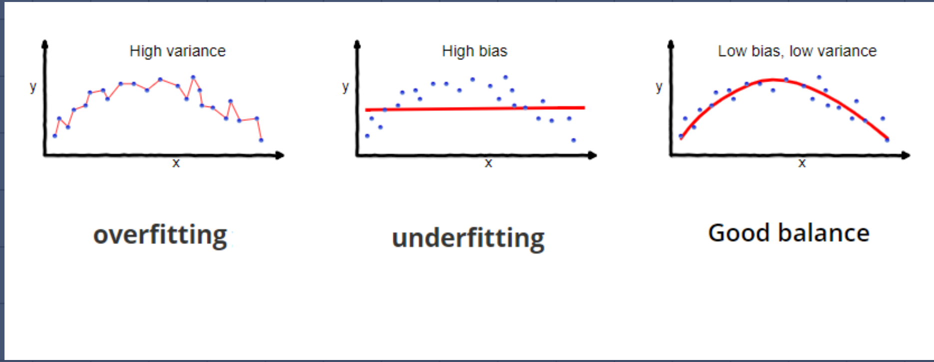


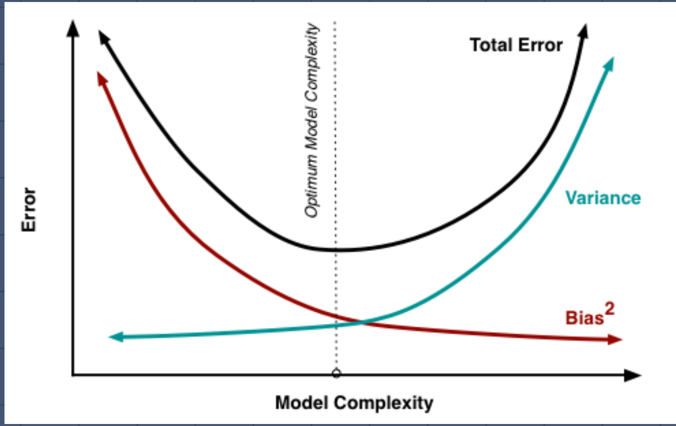
Comparing classification error of Train and Test sets depending on number of training samples

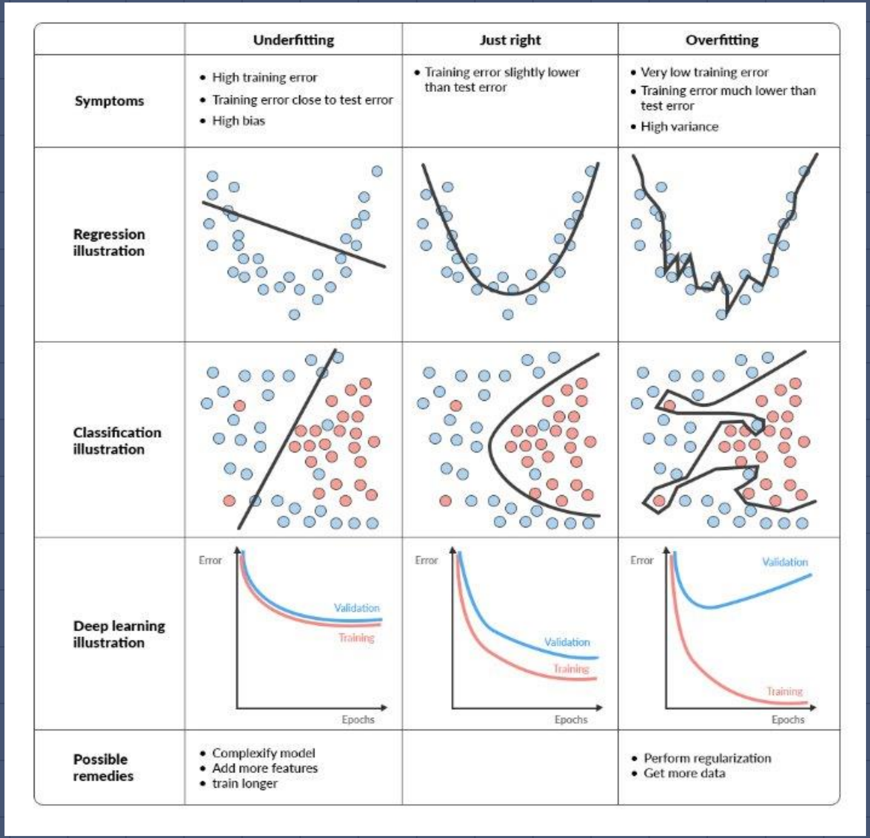




Intuitive understanding of bias–variance tradeoff







	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration	Error, Validation, Training, Epochs	Error, Validation, Training, Epochs	Error, Validation, Training, Epochs
Possible remedies	• Complexify model • Add more features • train longer		• Perform regularization • Get more data

Cross Validation Score

Evaluating model on the several splits (folds)

Splits the data to K folds

-> Fits the data on (K-1) folds

-> evaluates on remaining fold

```
from sklearn.model_selection import cross_val_score
print('Cross-validation (accuracy)', cross_val_score(clf, X, y, cv=5))
print('Cross-validation (AUC)', cross_val_score(clf, X, y, cv=5, scoring = 'roc_auc'))
print('Cross-validation (recall)', cross_val_score(clf, X, y, cv=5, scoring = 'recall'))
```

```
Out:
Cross-validation (accuracy) [0.91944444 0.98611111 0.97214485 0.97493036 0.96935933]
Cross-validation (AUC) [0.9641871  0.9976571  0.99372205 0.99699002 0.98675611]
Cross-validation (recall) [0.81081081 0.89189189 0.83333333 0.83333333 0.83333333]
```

Regression evaluation metrics

R-Squared

$$R^2 = 1 - \frac{SSE}{SST} \qquad \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

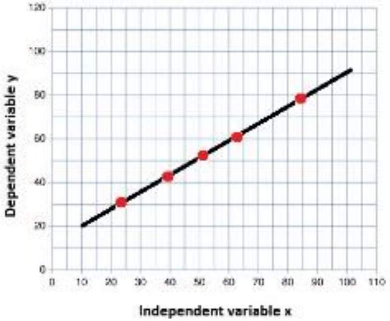
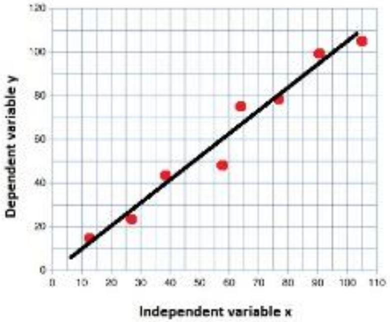
SSE - sum of squared error
SST - sum of squared total

R-Squared Indicates how better is my model compared to basic mean model.

The Range of **R-Squared** is 0 to 1.

- If R-Squared is close to 1 → It is much better model than mean model.
- If R-Squared is close to 0.5 → It Requires Tuning → 0.4, 0.6.
- If R-Squared is close to 0 → It is Poor Model → 0.1, 0.2 → Discard the Model / Change the Algorithm.

R2 interpretation

R^2 **Values**	**Interpretation**	**Graph**
$R^2 = 1$	All the variation in the y values is accounted for by the x values	
$R^2 = 0.83$	83% of the variation in the y values is accounted for by the x values	

R-Squared adjusted

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where

R^2 Sample R-Squared

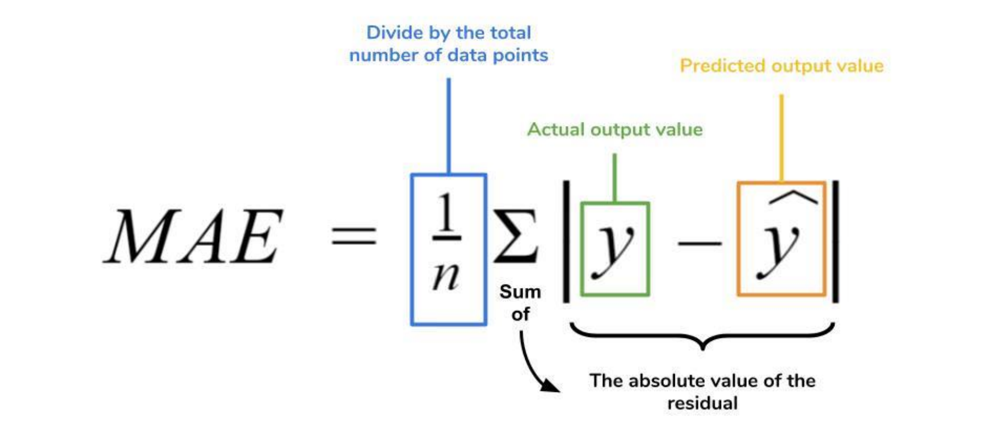
N Total Sample Size

p Number of independent variable

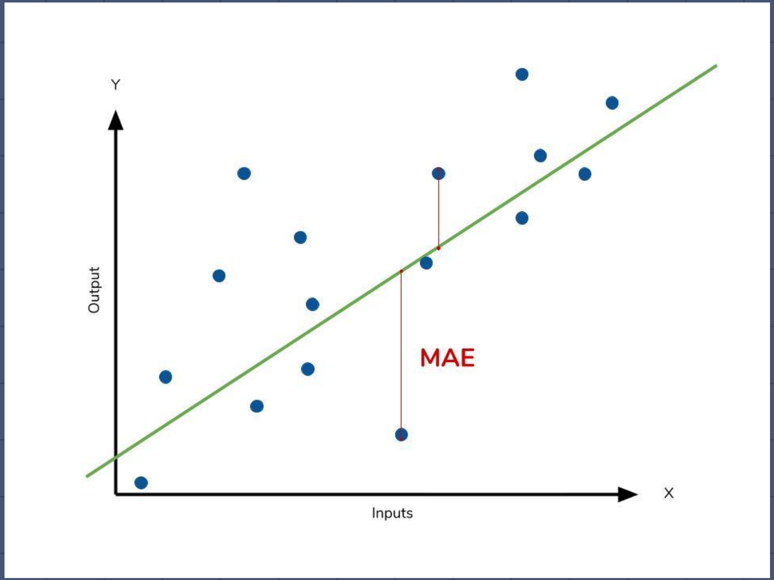
Although both r-squared and adjusted r-squared evaluate regression model performance, a key difference exists between the two metrics. The r-squared value always increases or remains the same when more predictors are added to the model, even if those predictors do not significantly improve the model's explanatory power. This issue can create a misleading impression of the model's effectiveness.

Adjusted r-squared adjusts the r-squared value to account for the number of independent variables(features) in the model. The adjusted r-squared value can decrease if a new predictor does not improve the model's fit, making it a more reliable measure of model accuracy. For this reason, the adjusted r-squared can be used as a tool by data analysts to help them decide which predictors to include.

MAE



MAE - measures the absolute difference between the model's predictions and the data. Each residual contributes equally to the total error, with larger errors contributing more to the overall error. A small MAE indicates good prediction performance, while a large MAE suggests that the model may struggle in certain areas.



MSE, RMSE

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

...while each residual in MAE contributes proportionally to the total error, the error grows quadratically in **MSE**. This ultimately means that outliers in our data will contribute to a much higher total error in the MSE than they would in the MAE

The **RMSE** is used to convert the error metric back into similar units as the original output, making interpretation easier. Like the MSE, the RMSE is also affected by outliers.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f_i - o_i)^2}$$

Home Task

Get practice with various metrics

for various datasets

for various models

Next Lesson

- Grid search
- Blight Fines Classification

Learn more

Applied Machine Learning in Python
https://www.coursera.org/learn/python-machine-learning

Sklearn.datasets.load_digits
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

Optical Recognition of Handwritten Digits Data Set
http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

sklearn.dummy.DummyClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

sklearn.metrics.roc_auc_score
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

sklearn.model_selection.cross_val_score
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html#sklearn.model_selection.cross_val_score

Learn more

Some more ML evaluation metrics for regression
https://www.appsilon.com/post/machine-learning-evaluation-metrics-regression

THANKS!

Any questions?