Introduction to Modeling in Python

Five College DataFest 2019
Quentin Dupupet (MassMutual DSDP)

“All models are wrong, but some are useful”

What is a model?

- A ***model*** is a simplified representation of a potentially complex real-world phenomenon
- Mathematically, a model is a function of form $y = f(X)$

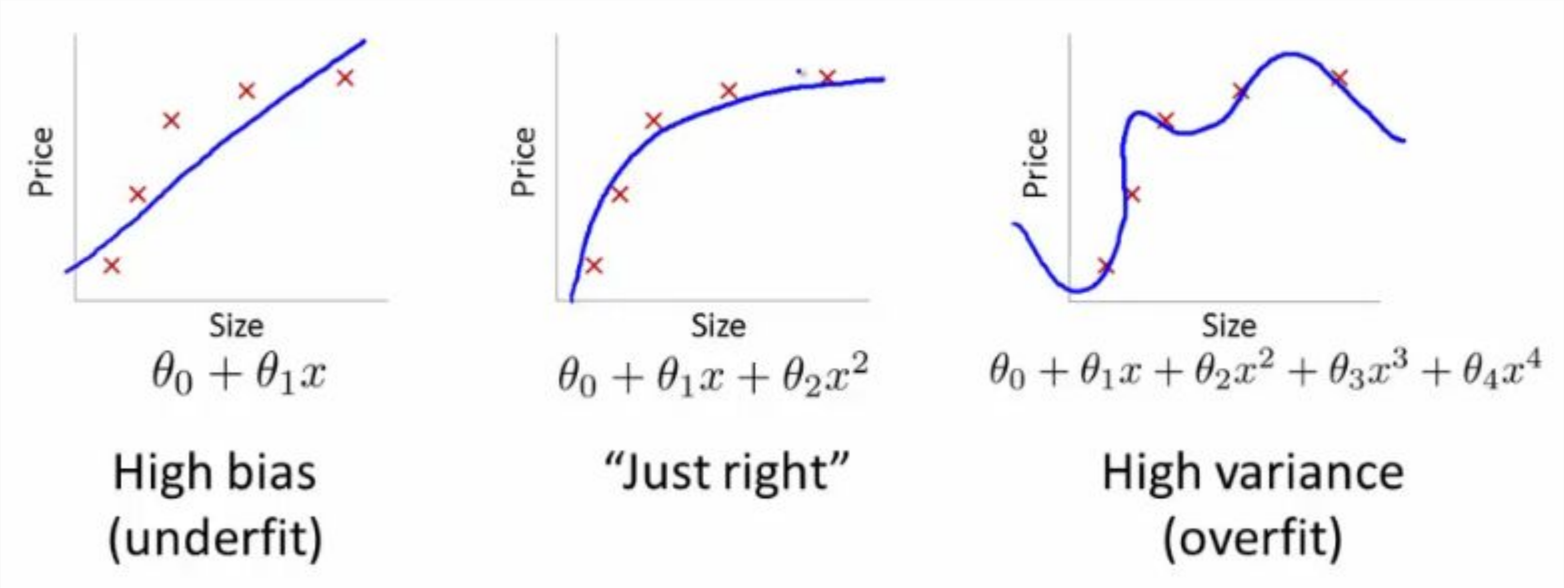
How does a model work?

- Models “learn” by associating a set of various observations of different feature inputs (e.g. weight, age) with a set of corresponding outputs (e.g. height) in a way such that an arbitrary ***cost function*** is minimized

Model performance

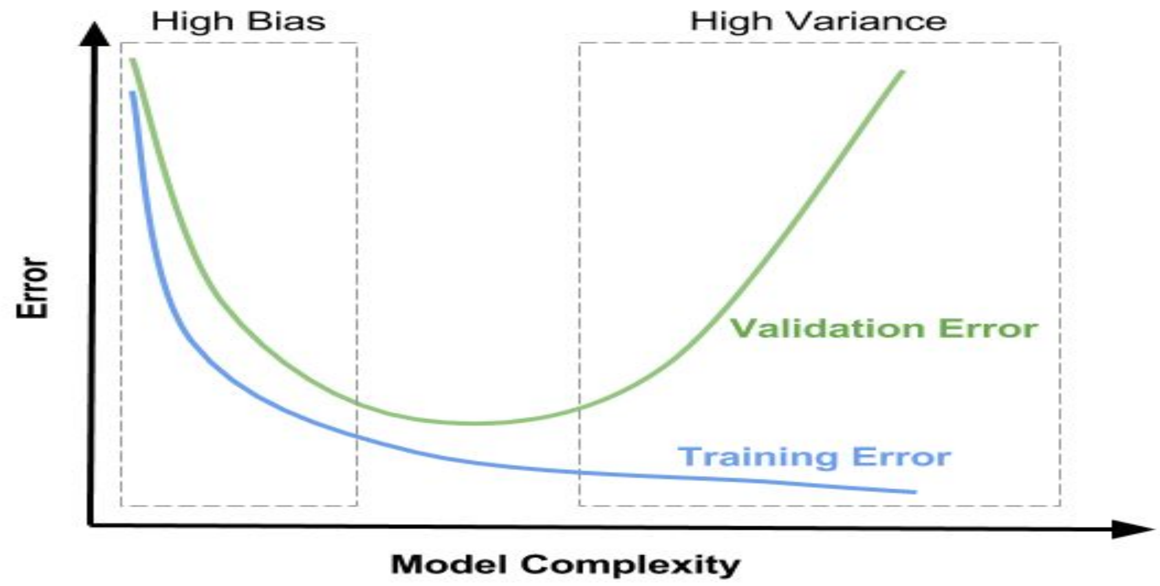
- Models are ultimately judged on their ability to generalize to never-before seen data (frequently known as the “*test set*”)
- A fundamental concept in modeling is the *bias-variance* tradeoff
  - ***Bias*** - error introduced in the specification of the form of the model (i.e. underfitting)
  - ***Variance*** - error introduced by inherent variability in the data itself (i.e. overfitting)

Bias-variance tradeoff



http://www.turingfinance.com/perils-optimization-in-investment-management/

Bias-variance tradeoff



https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/

Code Example: Loading Data

```
from sklearn.datasets import load_boston

data = load_boston()
df = pd.DataFrame(data.data, columns = data.feature_names)
df['target'] = data.target
df.head()
```

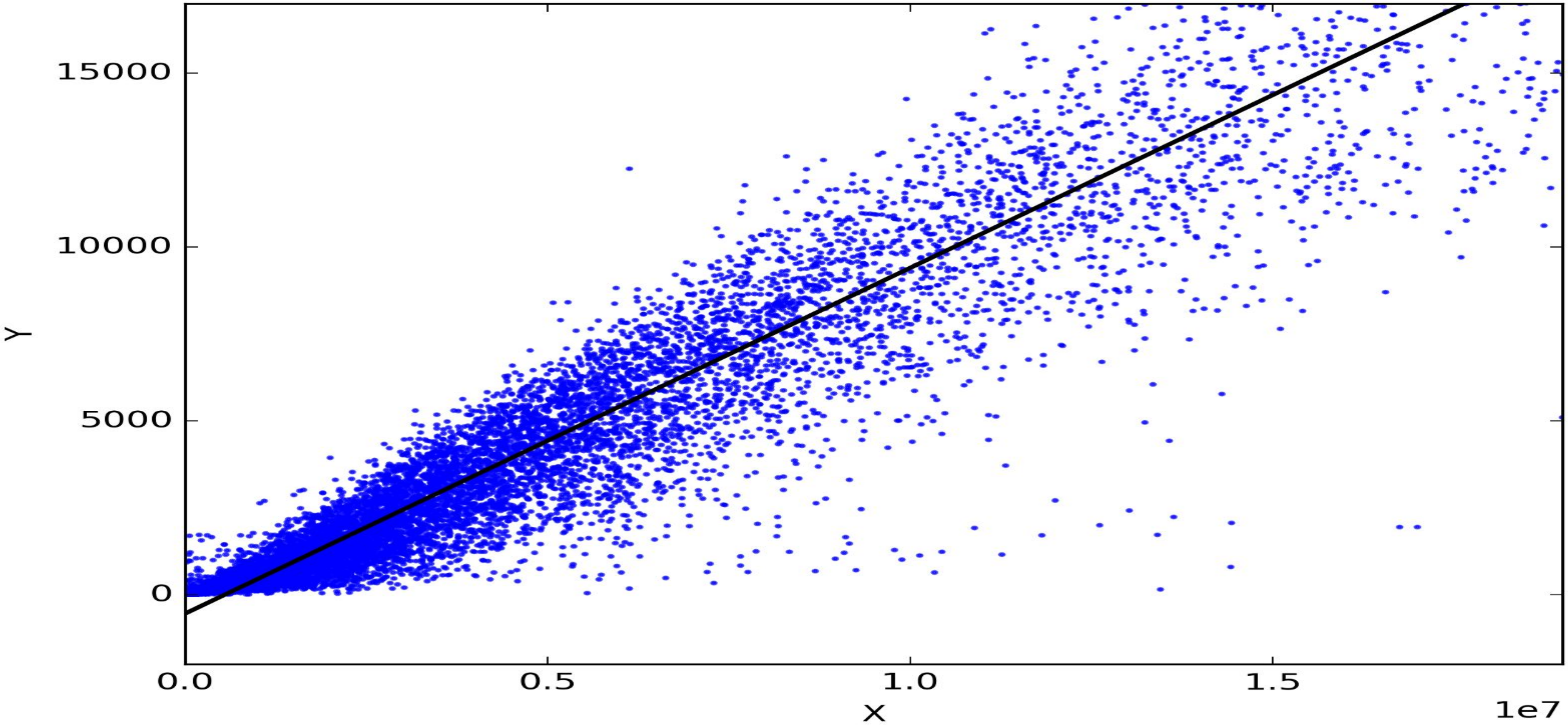
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

Train/test code

```
from sklearn.model_selection import train_test_split

X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(379, 13) (379,) (127, 13) (127,)
```

Regression



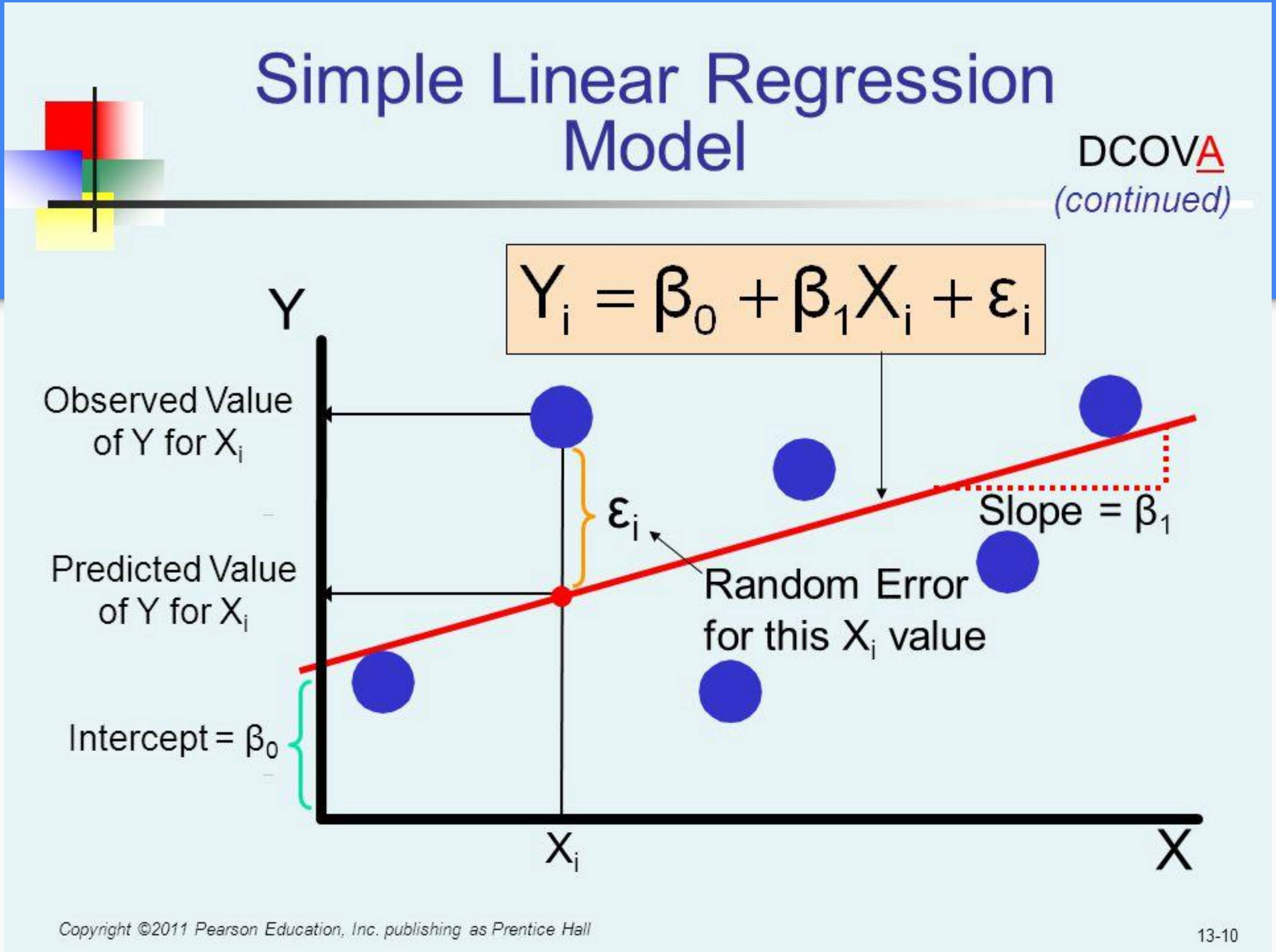
https://medium.com/@amarbudhiraja/ml-101-linear-regression-tutorial-1e40e29f1934

Regression background

"***Regression***" is the act of predicting a continuous quantity (e.g. the price of something)

Linear regression

An common regression model is ***linear regression***, which fits an (N-1)-dimensional hyperplane to an MxN numeric matrix of observations such that distance between the hyperplane and all observations is minimized



https://github.com/tirthajyoti/R-stats-machine-learning/blob/master/README.md

Linear regression code - fitting the model

```
import numpy as np
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

print(
    np.round(lin_reg.coef_,1),
    '\n',
    np.round(lin_reg.intercept_, 1)
)
```

```
[-0.1  0.  -0.2  0.  -0.   0.1 -0.3 -0.2 -0.   0.1 -0.2 -0.3 -0. ]
 3.5
```

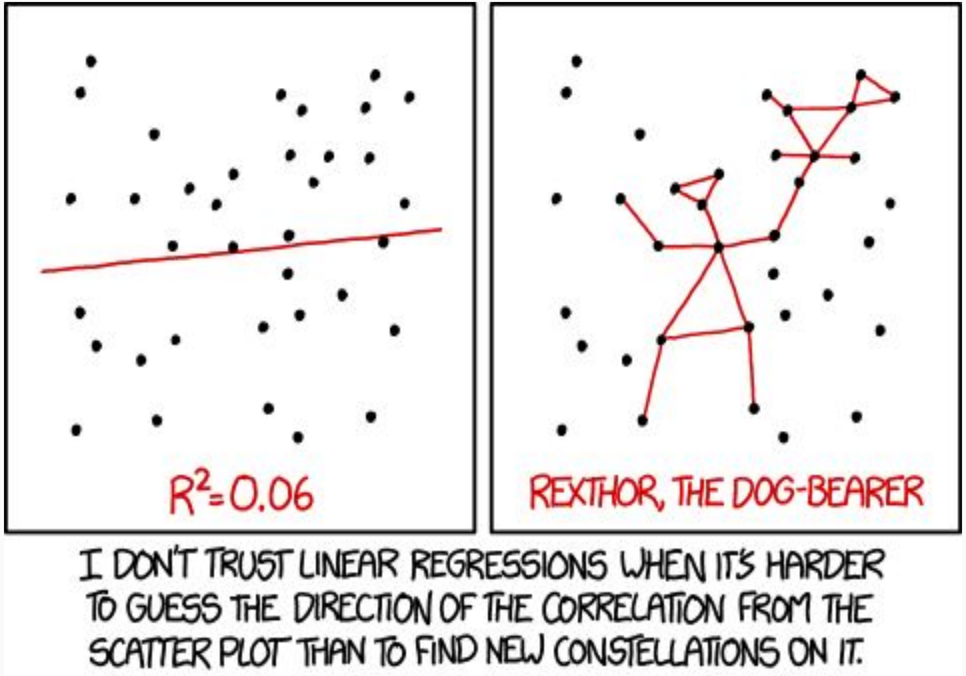
$$y = -0.1(CRIM) - 0.2(INDUS) + 0.1(RM) - 0.3(AGE) - 0.2(DIS) + 0.1(TAX) - 0.2(PTRATIO) - 0.3(B) + 3.5$$

Linear regression code - evaluation

```
from sklearn.metrics import mean_squared_error
y_pred = lin_reg.predict(X_test)

print('R^2 Score:', lin_reg.score(X_test, y_test))
print('Mean squared error: ', mean_squared_error(y_test, y_pred))
```

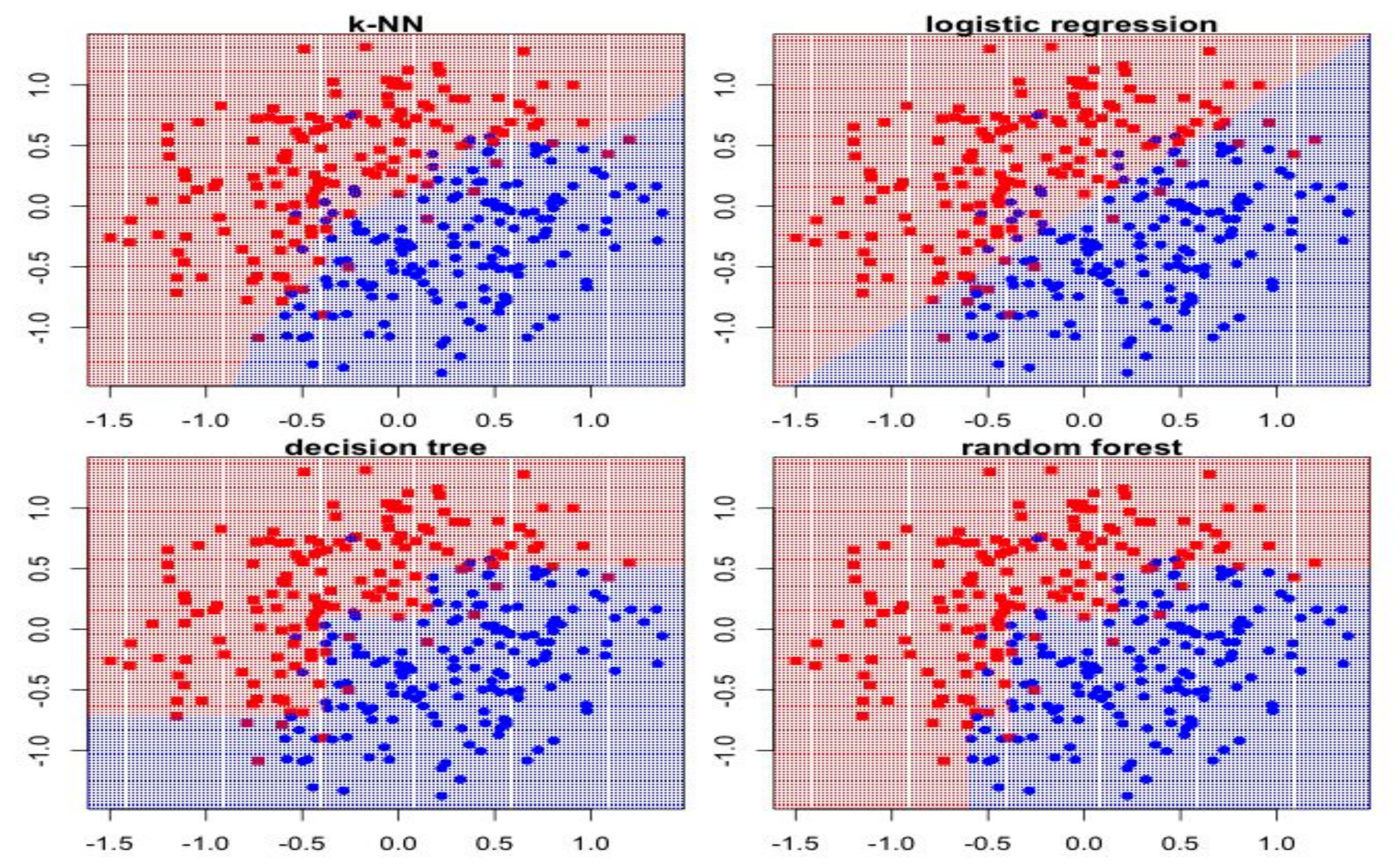
```
R^2 Score: 0.8850825151531964
Mean squared error:  0.06843974653098528
```

Relevant XKCD



https://www.explainxkcd.com/wiki/index.php/1725:_Linear_Regression

Classification



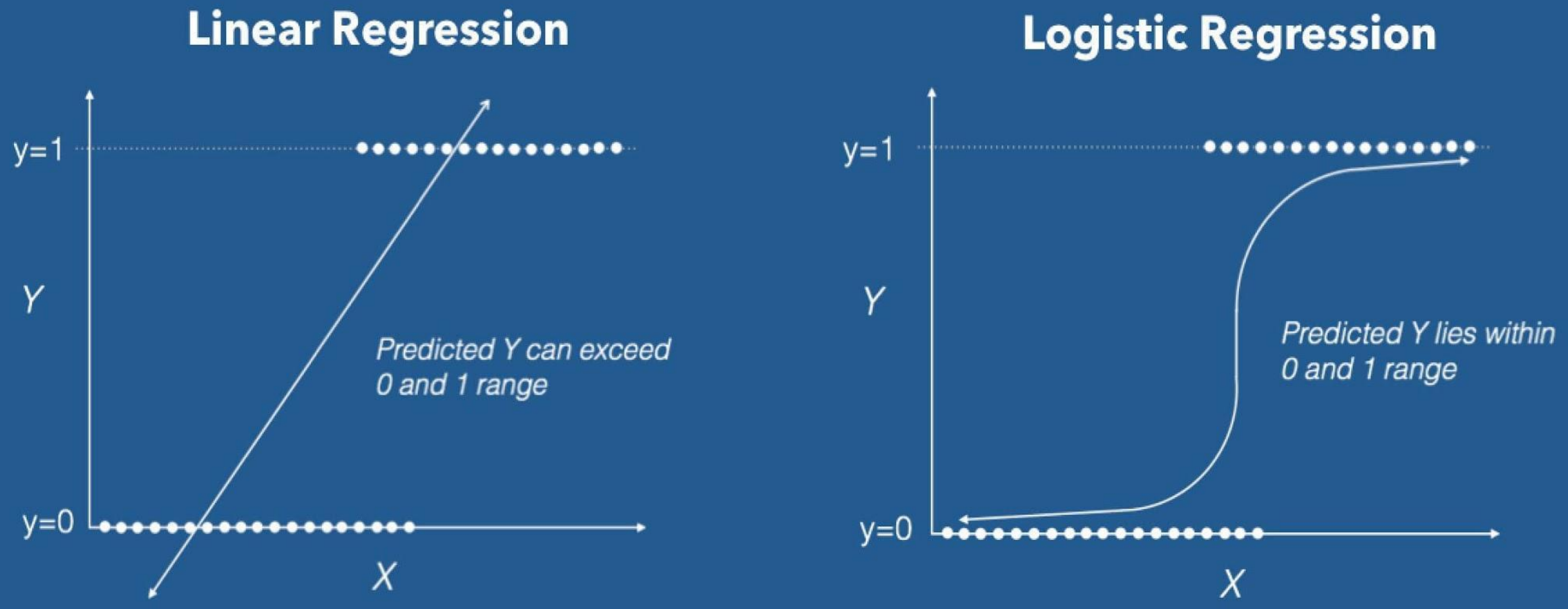
https://www.oreilly.com/library/view/data-analysis-with/9781788393720/7ef9ff0a-000c-44e1-ad6a-d352e0744c74.xhtml

Classification background

In ***classification***, our interest is a discrete class rather than a continuous quantity (e.g. a label indicating whether a sale was made or not), and our goal is to find a ***decision boundary*** that accurately separates classes

Logistic regression

- A common binary classification model is ***logistic regression***, which outputs the probability of each observation belonging to the positive class
    - Like linear regression, it consists of a linear set of coefficients which are fed into a link function ensuring they output a real number between 0 and 1



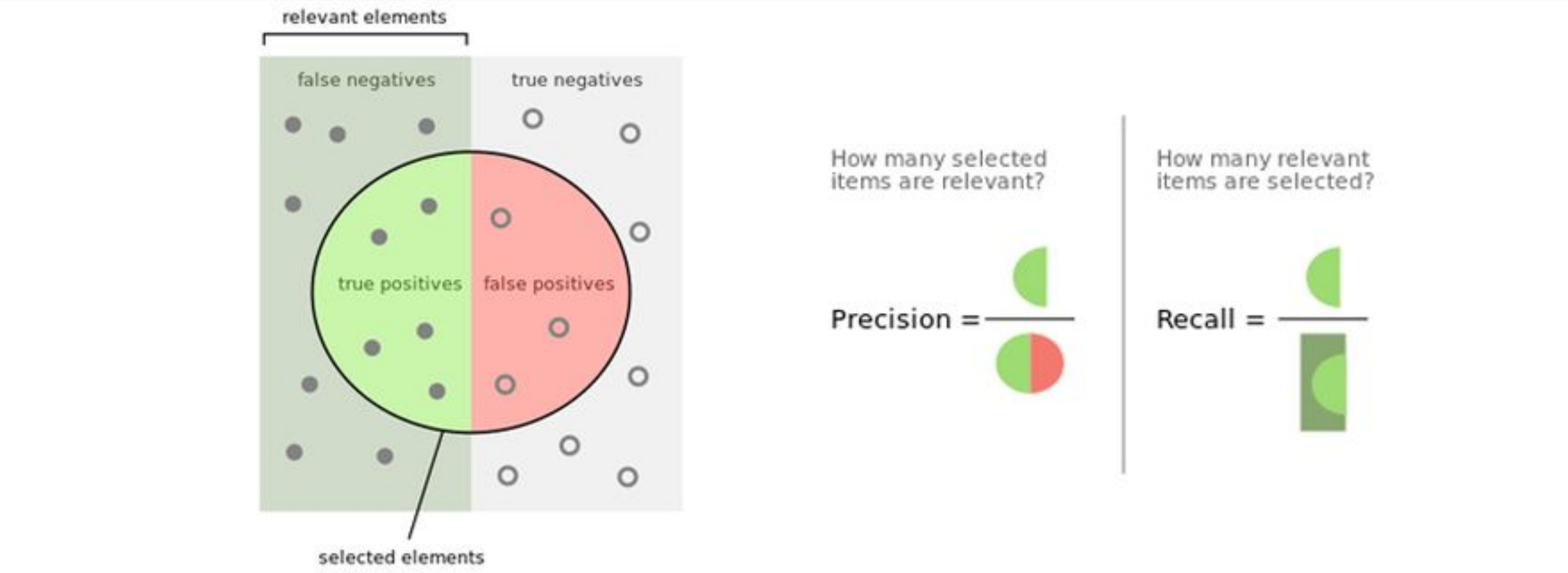
https://medium.com/greyatom/logistic-regression-89e496433063

Classification evaluation - confusion matrix

		True class: Positive	True class: Negative	Measures
Predicted class	Positive	True positive *TP*	False positive *FP*	Positive predictive value (PPV) $\frac{TP}{TP+FP}$
	Negative	False negative *FN*	True negative *TN*	Negative predictive value (NPV) $\frac{TN}{FN+TN}$
Measures		Sensitivity $\frac{TP}{TP+FN}$	Specificity $\frac{TN}{FP+TN}$	Accuracy $\frac{TP+TN}{TP+FP+FN+TN}$

https://www.researchgate.net/figure/Contingency-matrix-and-measures-calculated-based-on-it-2x2-contigency-table-for_fig4_230614354

Classification evaluation - precision/recall



https://www.digital-mr.com/blog/view/precision-and-recall-in-social-listening

Classification Code

```
from sklearn.datasets import load_wine

wine = load_wine()
wine_df = pd.DataFrame(wine.data, columns = wine.feature_names)
wine_df['target'] = wine.target
wine_df.head()
```

ıesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0

```
X = wine.data
y = wine.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, stratify = y)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(133, 13) (133,) (45, 13) (45,)
```

Classification code - fitting the model

```python
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

print(
    np.round(log_reg.coef_, 2),
    '\n',
    np.round(log_reg.intercept_)
)
```

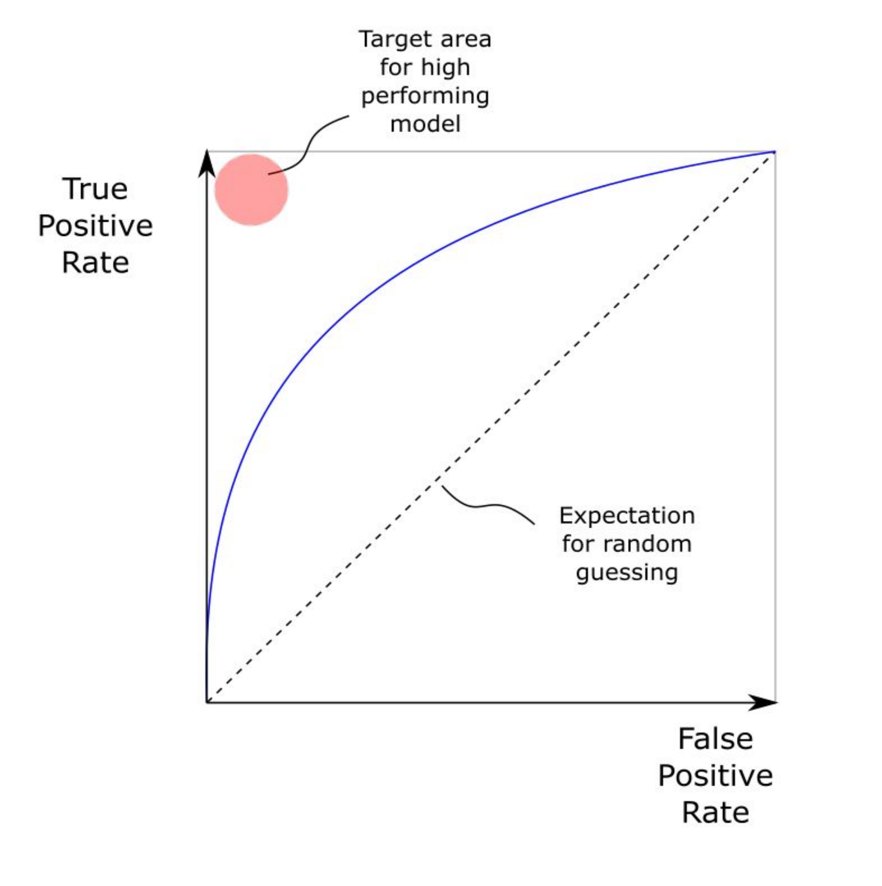
```
[[-0.57  0.78  1.03 -0.58 -0.02 -0.04  1.23  0.05 -0.44 -0.   -0.15  0.82
   0.02]
 [ 0.68 -1.04 -0.7   0.26  0.01  0.32  0.52  0.21  0.45 -1.87  0.73  0.15
  -0.01]
 [-0.08  0.72 -0.08  0.06  0.01 -0.61 -1.49 -0.01 -0.64  1.33 -0.41 -1.26
  -0.  ]]
 [-0.  0. -0.]
```

Classification code - evaluation

```
from sklearn.metrics import confusion_matrix, classification_report

print('Accuracy: ', log_reg.score(X_test, y_test))
y_pred = log_reg.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred)
print('\nConfusion Matrix:\n',
      pd.DataFrame(
          confusion_matrix(y_test, y_pred),
          columns = ['pred: 0', 'pred: 1', 'pred: 2'],
          index = ['true: 0', 'true: 1', 'true: 2']),
      '\n\nClassification Report:\n',
      classification_report(y_test, y_pred))
```

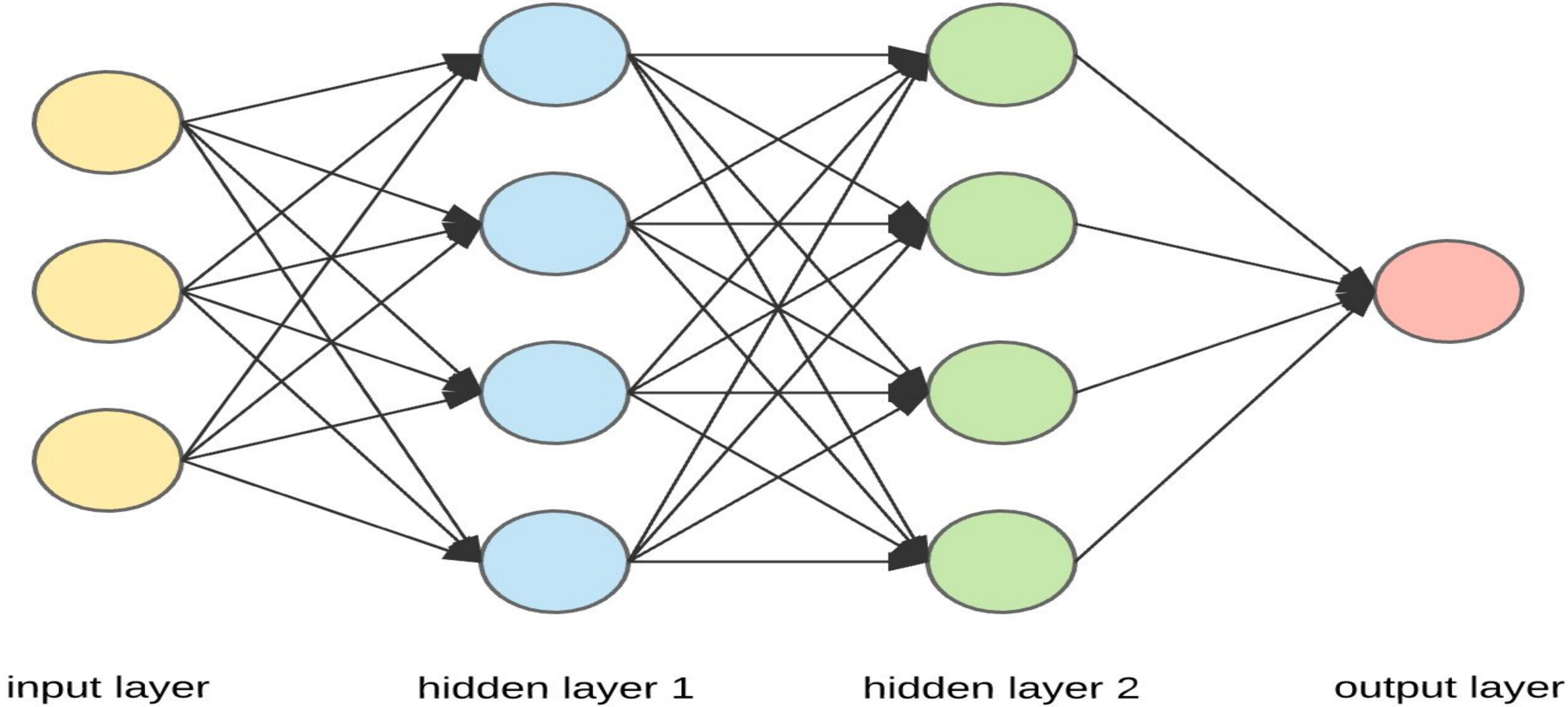
```
Accuracy:  0.9555555555555556

Confusion Matrix:
          pred: 0  pred: 1  pred: 2
true: 0       15        0        0
true: 1        0       16        2
true: 2        0        0       12

Classification Report:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        15
          1       1.00      0.89      0.94        18
          2       0.86      1.00      0.92        12

avg / total       0.96      0.96      0.96        45
```

ROC curve



https://deparkes.co.uk/2018/02/16/the-roc-curve/

Advanced Methods

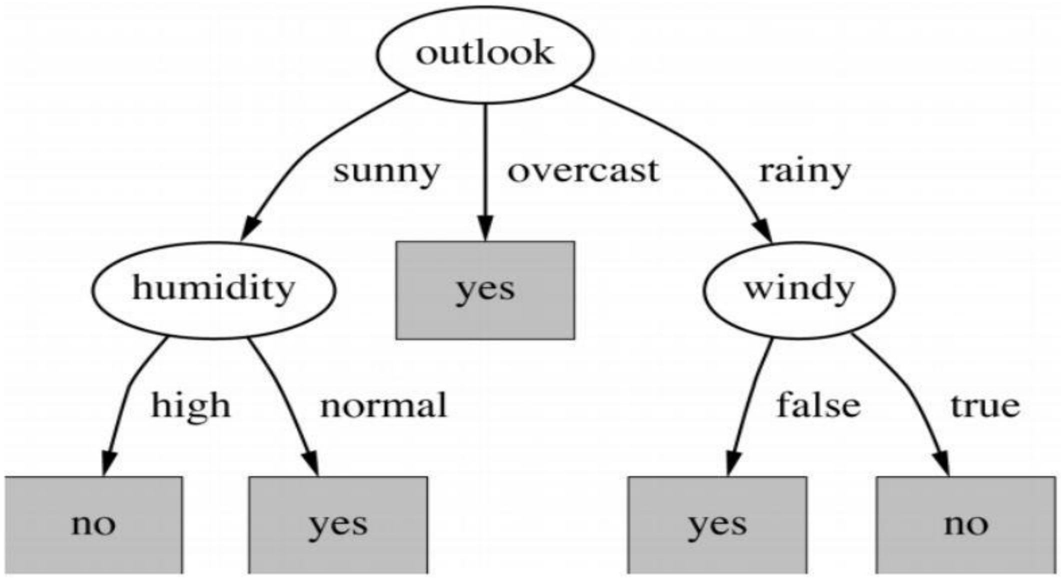


https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Decision tree background

- ***Decision trees*** attempt to split observations into distinct groupings based on feature values such that bin purity is maximized, ultimately creating a binary tree in the process
    - The predicted class will then be the maximally-represented class in the leaf nodes for classification, or an average of continuous values for a regression

Decision tree (playing tennis)

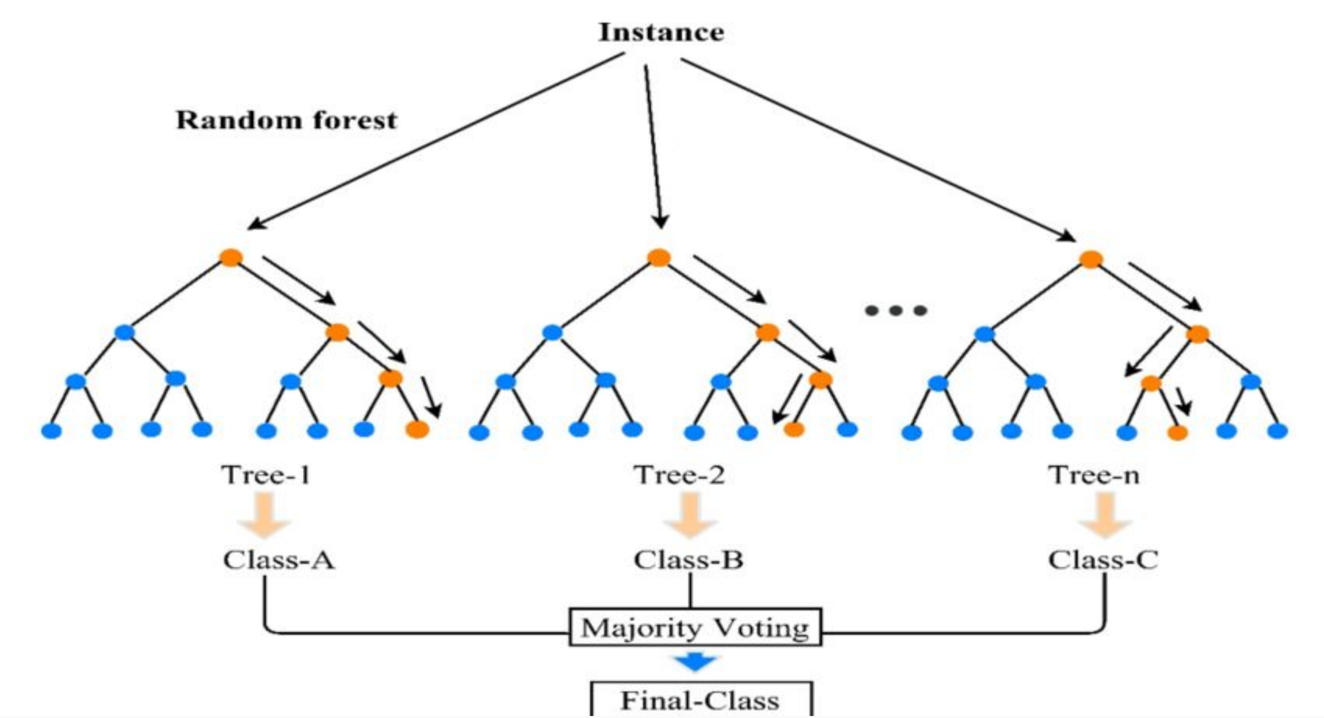


https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1

Random forest background

In order to reduce variance and chance of overfitting, a ***random forest*** is an average of the results of many decision trees splitting on random subsets of features

Random forest example



https://www.researchgate.net/figure/Random-Forests-Naive-Bayes-NB-NB-approaches-are-a-family-of-simple-probabilistic_fig2_326722598

Random forest code example: Boston

```
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor(max_depth = 4)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)
print(rf_reg.feature_importances_)
print('\nR^2 Score:', rf_reg.score(X_test, y_test))
print('Mean squared error: ', mean_squared_error(y_test, y_pred_rf))
```

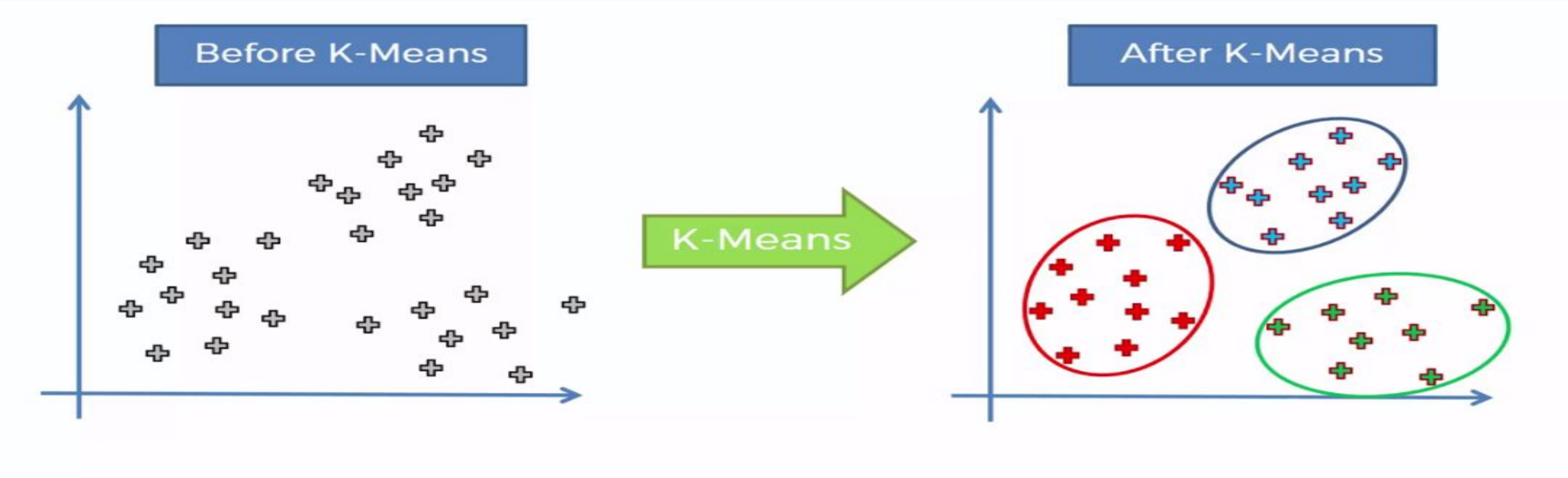
```
[0.06543473 0.00104719 0.00340049 0.         0.00376146 0.
 0.38850392 0.         0.00343326 0.08661245 0.         0.22357216
 0.22423434]

R^2 Score: 0.9059701492537314
Mean squared error:  0.056
```

Additional types of models

- Regression - regularized linear regression (LASSO, Ridge, Elastic Net), polynomial regression/splines
- Classification - Naive Bayes, linear/quadratic discriminant analysis
- Regression/classification - support vector machines, k-nearest neighbors, XGBoost
- Survival analysis, time series, mechanistic modeling, causal inference, many more...

Unsupervised learning

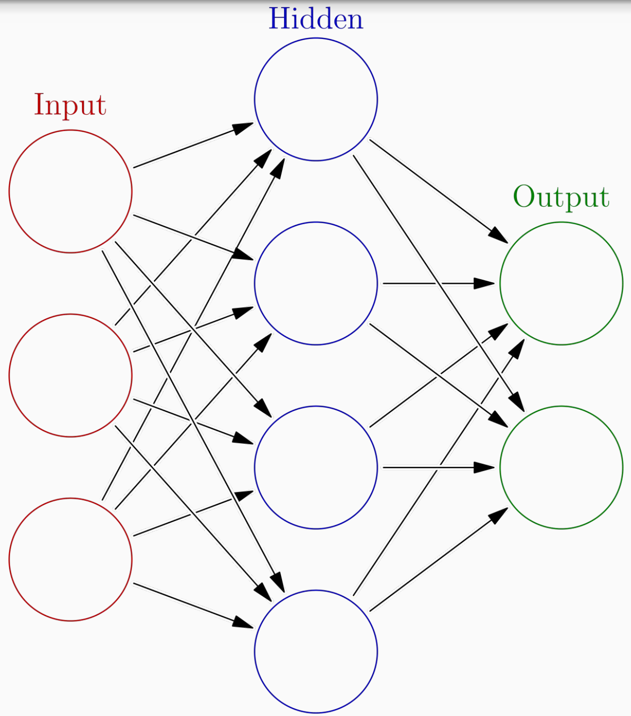
- Infer *y* from the data (no explicit optimization criteria)



https://towardsdatascience.com/k-means-clustering-identifying-f-r-i-e-n-d-s-in-the-world-of-strangers-695537505d

Deep learning

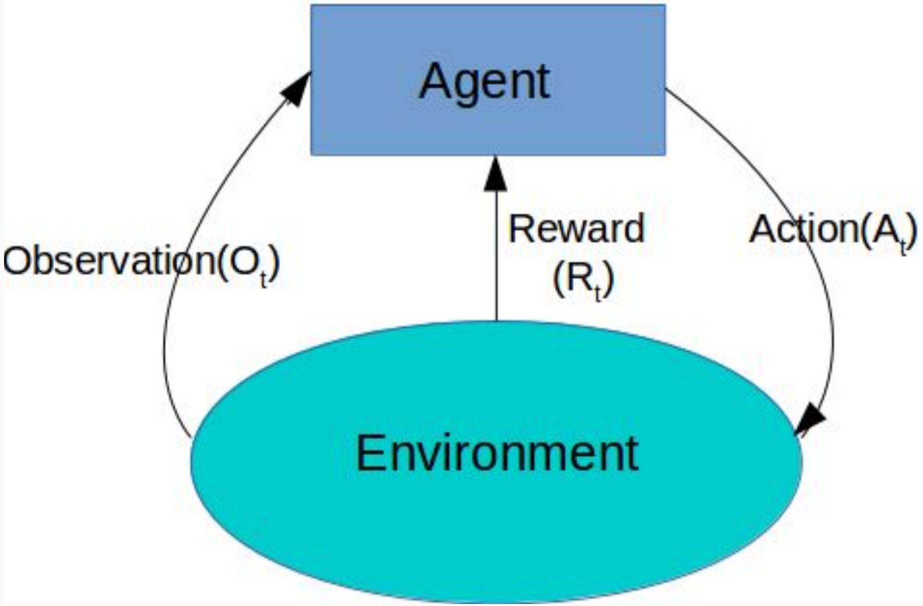
- Send input through a nested series of “hidden” activation functions before reaching the output



https://en.wikipedia.org/wiki/Artificial_neural_network

Reinforcement learning

- Choose the best actions in a system where many are possible in order to optimize some reward function over time



https://becominghuman.ai/components-of-an-rl-agent-and-its-application-on-snake-1b3b6c8e1de5

Additional resources

- Linear regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- Logistic regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- More methods and explanations:https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html

Exercises

Link to respective Jupyter Notebook / RMD files here