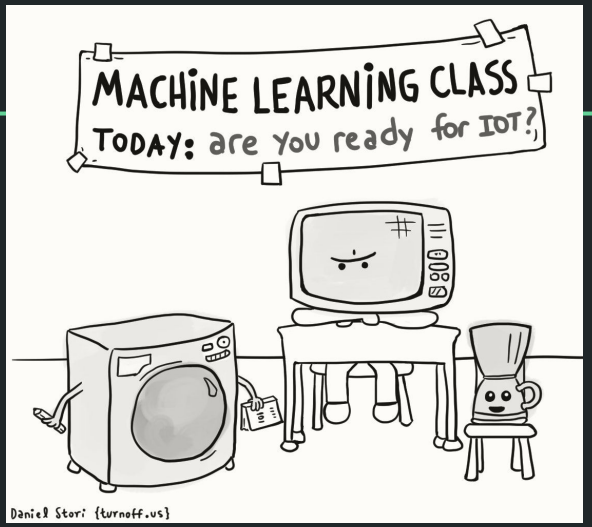
MANCHESTER 1824
The University of Manchester

JACS: Intro to Machine Learning

Hayden Rampadarath



13th November 2017
https://github.com/hrampadarath/JACS_ML_Tutorial

Article

https://www.nature.com/articles/nature24270

Mastering the game of Go without human knowledge

David Silver✉, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel & Demis Hassabis

“Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher … AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

What is Machine Learning?

Machine learning is ...

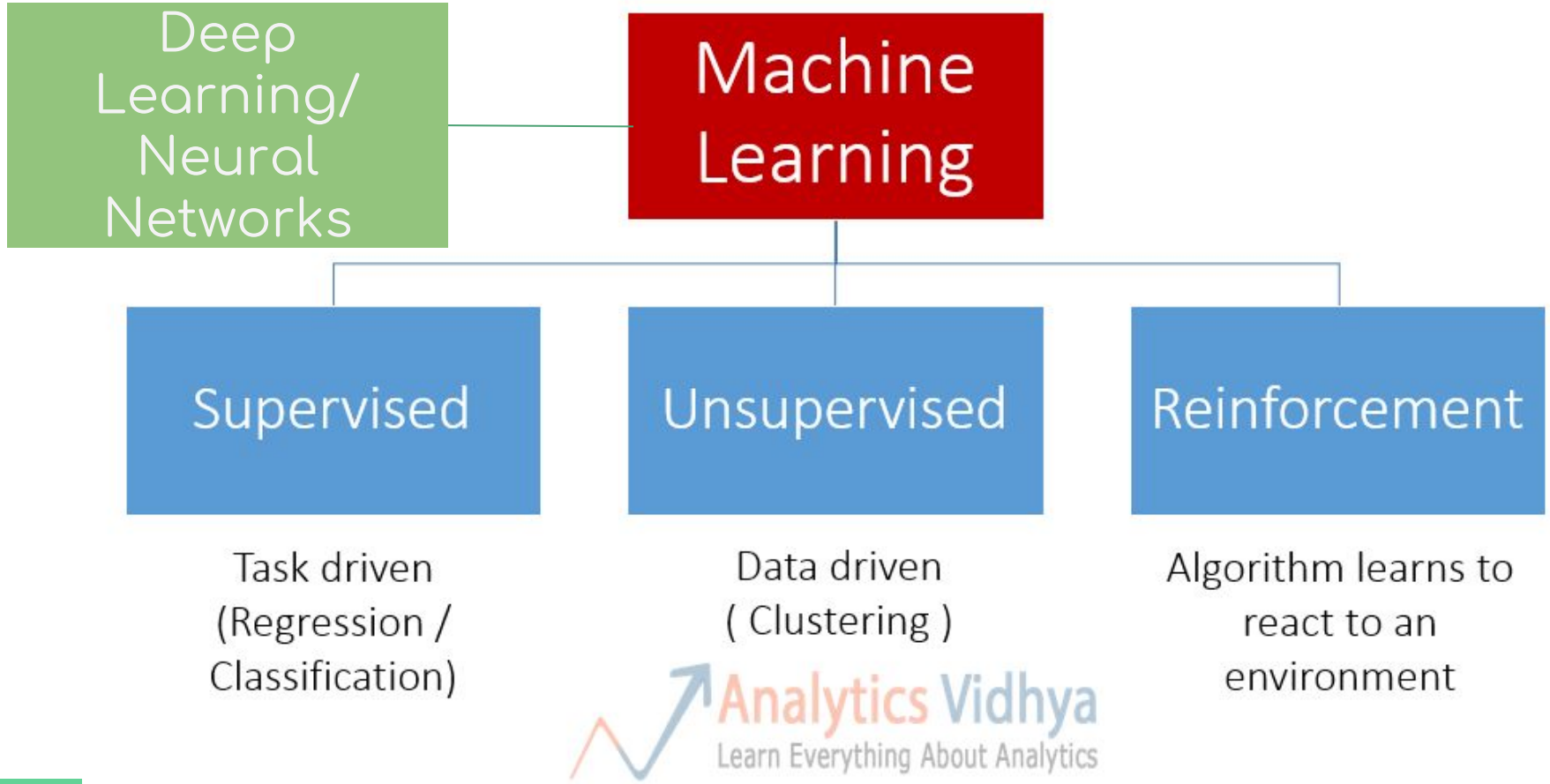
a field of computer science that gives computers the ability to learn without being explicitly programmed. - Wikipedia ML page

a subfield of computer science and artificial intelligence which deals with building systems that can learn from data, instead of explicitly programmed instructions. - AnalyticsVidya.com

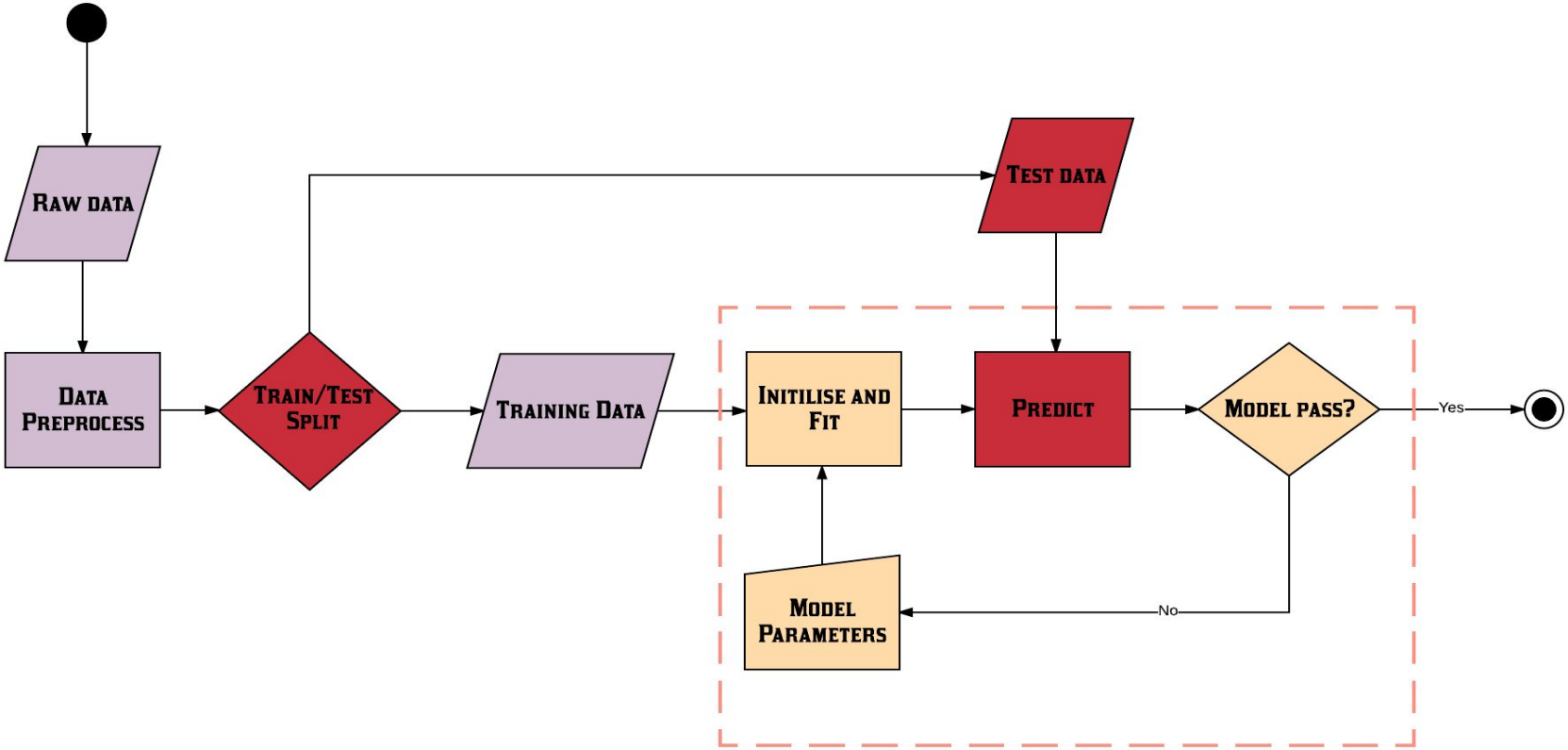
Statistical Modelling is ...

a subfield of mathematics which deals with finding relationship between variables to predict an outcome - AnalyticsVidya.com

Types of Machine Learning

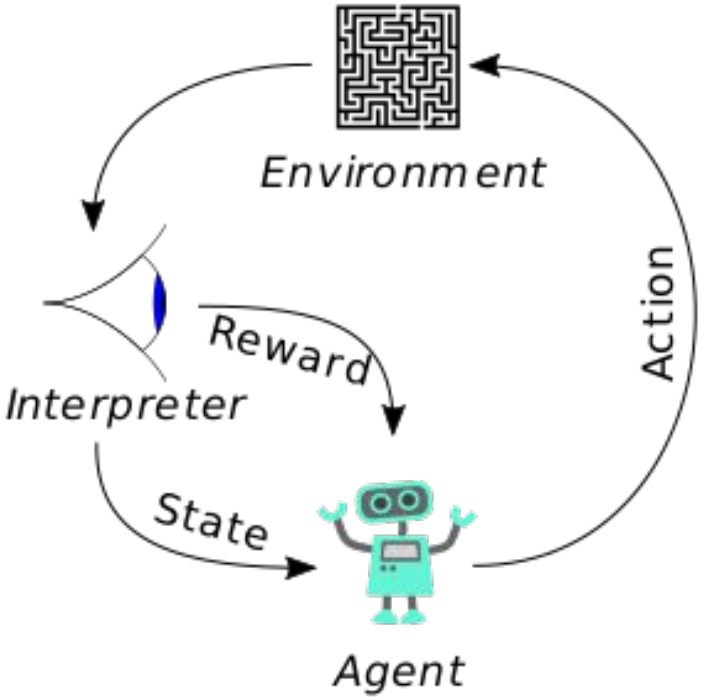


Supervised Learning - Most Common



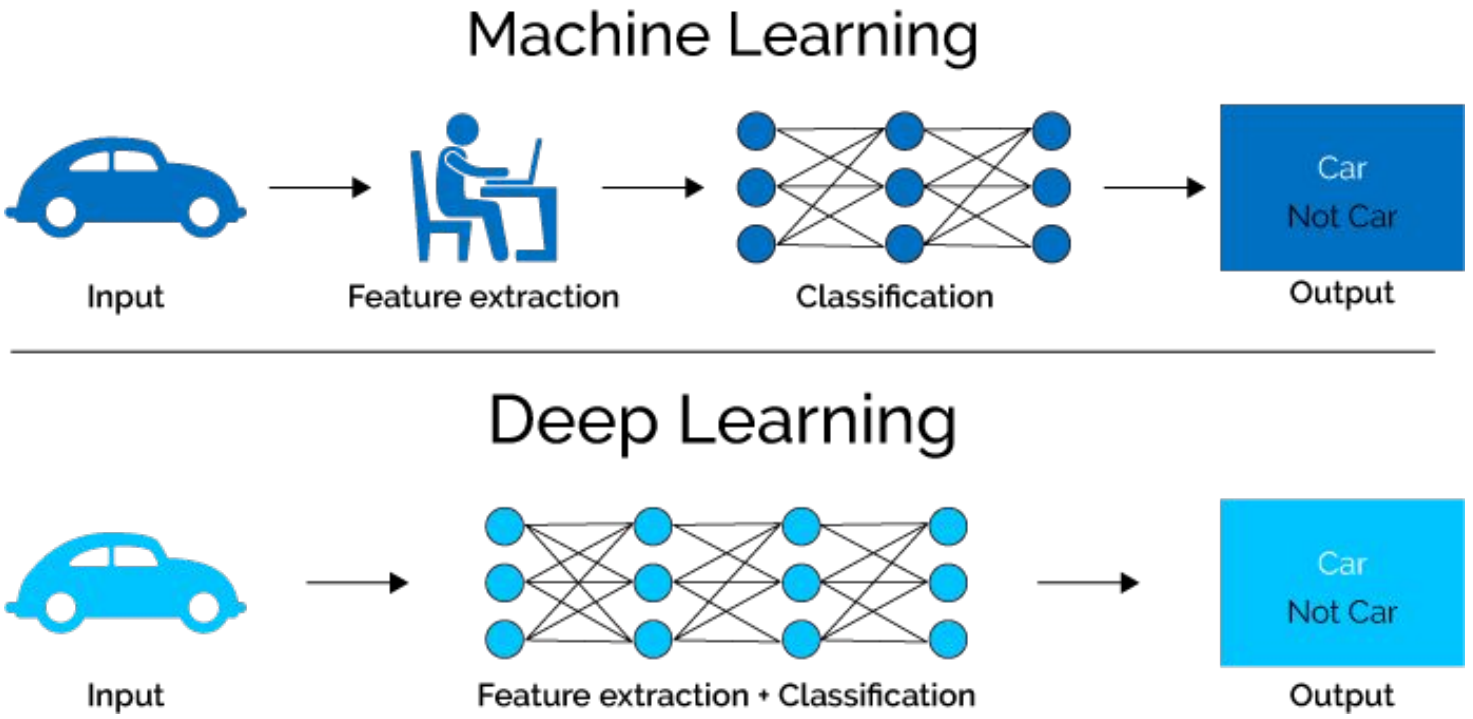
Reinforcement Learning

Diagram showing the components in a typical Reinforcement Learning (RL) system. An agent takes actions in an environment which is interpreted into a reward and a representation of the state which is fed back into the agent.



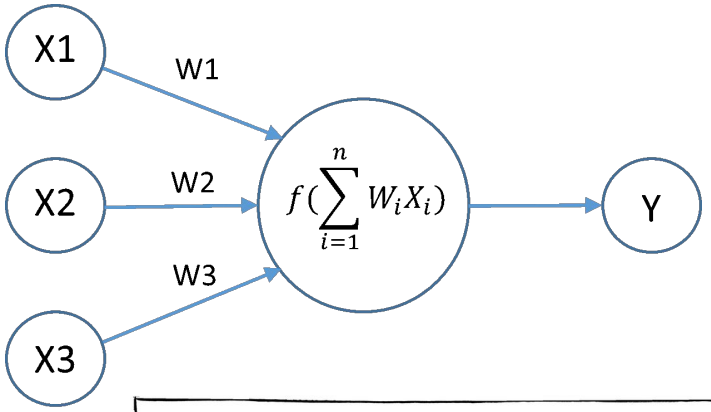
From Wikipedia and incorporates other CC0 work:
https://openclipart.org/detail/202735/eye-side-view
https://openclipart.org/detail/191072/blue-robot and
https://openclipart.org/detail/246662/simple-maze-puzzle

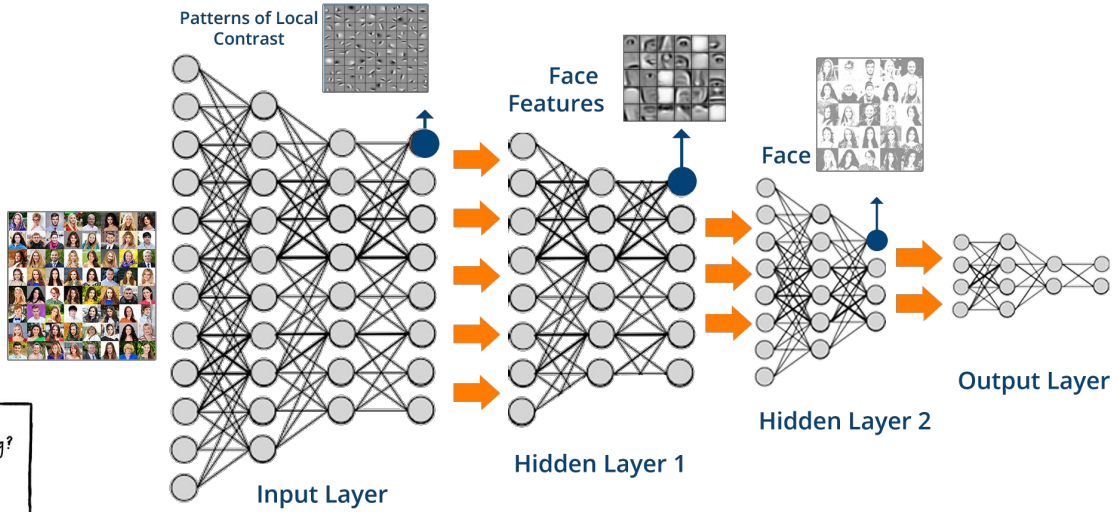
Deep Learning

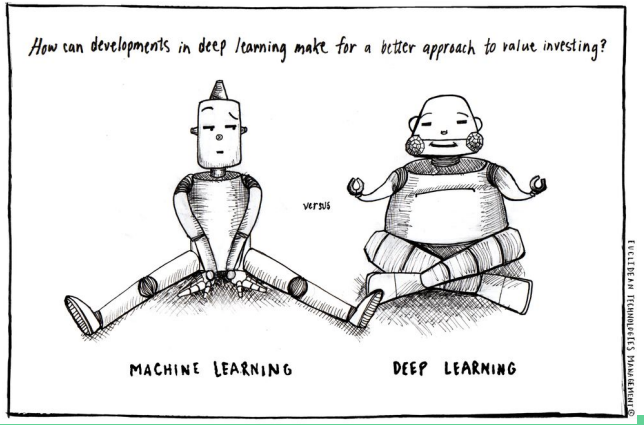


Deep Learning

Simple Neuron



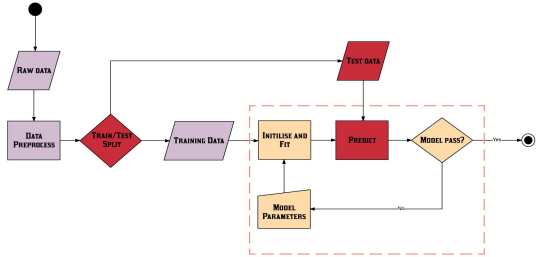




Applying Machine Learning: Libraries

- Supervised/Unsupervised/Reinforcement Learning:
  - Scikit-Learn: http://scikit-learn.org/stable/index.html
- Deep Learning:
  - Keras: https://keras.io/ (API)
  - TensorFlow: https://www.tensorflow.org/ (Engine)
  - Theano: http://www.deeplearning.net/software/theano/ (Engine)

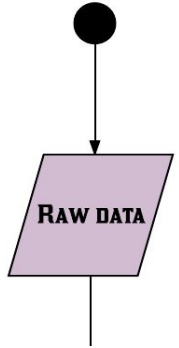
Applying Machine Learning: Tutorial



- A quick tutorial on using Python Scikit-Learn to analyse data from the Lending Club.
- The data was obtained from Kaggle [1]
- Complete loan data for all loans issued through the 2007-2015, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information.
- Cleaned and reduced for this tutorial.
- Follow using your favourite IDE or with Jupyter Notebook

[1] https://www.kaggle.com/wendykan/lending-club-loan-data)

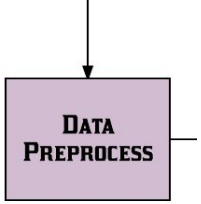
Basic ML Tutorial: 1. Load Raw Data and inspect




```
> import pandas as pd
> import numpy as np

# load data
> df = pd.read_csv('loans_data_cleaned.csv')
#inspect with either df.head() or df.info()
# look for null values. There should be none as I cleaned it before
>df.isnull().sum()
```

Basic ML Tutorial: 2. Some Preprocessing (1)

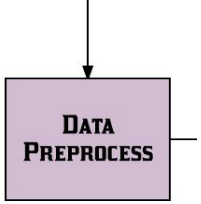


The goal of this exercise is to see whether we can predict someone will not pay off their loan or have become very delinquent on their payment. In this dataset it is in the 'loan_status' column.

```
>df.loan_status.unique() or print(df.loan_status.unique())
array(['Fully Paid', 'Late (31-120 days)', 'Late (16-30 days)',
       'Charged Off', 'In Grace Period', 'Default', 'Issued'], dtype=object)
```

The term **'Charged Off'** is the declaration by a creditor (usually a credit card account) that an amount of debt is unlikely to be collected. This occurs when a consumer becomes severely delinquent on a debt. Traditionally, creditors will make this declaration at the point of six months without payment. Thus we will be using this as our Class 1 or the positive case.

Basic ML Tutorial: 2. Some Preprocessing (2)



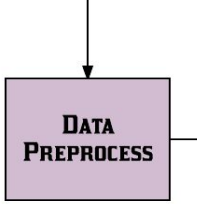
#set 'Charged Off' to 1 and the rest to 0

```
>mask = (df.loan_status == 'Charged Off')
>df['target'] = 0
>df.loc[mask,'target'] = 1
>target = df['target'] # here we have defined the target or the y - values
```

#The features or the 'X-values' will be all columns minus the 'target and loan_status' columns

```
>df = df.drop(['loan_status','target'],axis=1)
>print('Shape of features = {}, and of y-values = {}'.format(np.shape(df), np.shape(target)))
Shape of features = (209541, 38), and of y-values = (209541,)
```

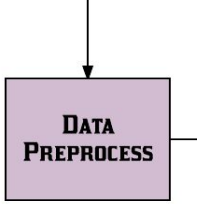
Basic ML Tutorial: 2. Some Preprocessing (3)



#Before we could start the ML part we need to take care of the categorical features.
#first separate the categorical features from continuous columns

```
>loan_categorical = df.select_dtypes(include=['object'], exclude=['float64','int64'])
>features = df.select_dtypes(include=['float64','int64'])
>print('Numerical features: \n{}'.format(list(features)))
>print('Categorical features: \n{}'.format(list(loan_categorical)))
```

Basic ML Tutorial: 2. Some Preprocessing (4)



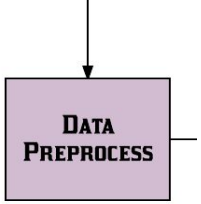
#One of the true power of ML and sklearn is the ability to use categorical data to train the model. But to do we need to convert to numerical data

```
>for col in list(loan_categorical):
        dummy = pd.get_dummies(loan_categorical[col])
        features = pd.concat([features,dummy],axis=1)

>print('Shape of features = {}, and of y-values = {}'.format(np.shape(features),
np.shape(target)))
```

Shape of features = (209541, 865), and of y-values = (209541,)

Basic ML Tutorial: 2. Some Preprocessing (4)

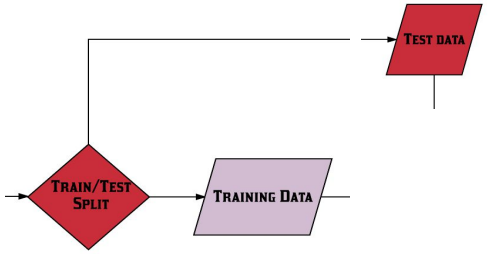


>features.head()

Out[23]:

inual_inc	dti	delinq_2yrs	inq_last_6mths	open_acc	...	Sep-2012	Sep-2013	Sep-2014	Sep-2015	experienced	inexp	intermed	new	seasoned	unknown
40000.0	16.94	0.0	0.0	7.0	...	0	0	0	0	1	0	0	0	0	0
25000.0	24.68	0.0	0.0	5.0	...	0	0	0	0	1	0	0	0	0	0
325000.0	18.55	0.0	1.0	15.0	...	0	0	0	0	0	0	1	0	0	0
100000.0	22.18	0.0	0.0	14.0	...	0	0	0	0	1	0	0	0	0	0
33000.0	15.75	0.0	1.0	9.0	...	0	0	0	0	0	0	0	1	0	0

Basic ML Tutorial: 2. Train/Test Split



- A fundamental part of ML is the Train/Test split.
- You do not want to use all your data to develop a model, as you can overfit on the data to create a perfect model for that data with no generalisation!
- Advisable to “hold” back some of the data (30%) to test your models

```
>from sklearn.model_selection import train_test_split
>X_train, X_test, y_train, y_test = train_test_split(features,target, random_state=0)
```

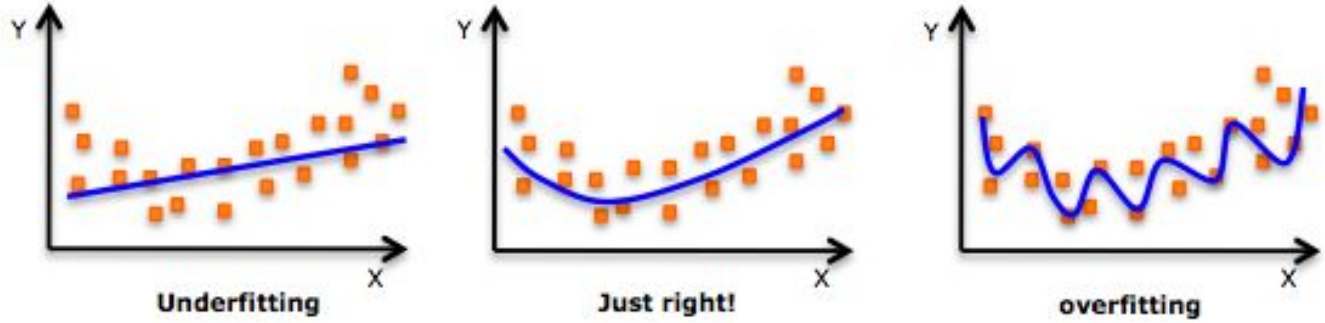
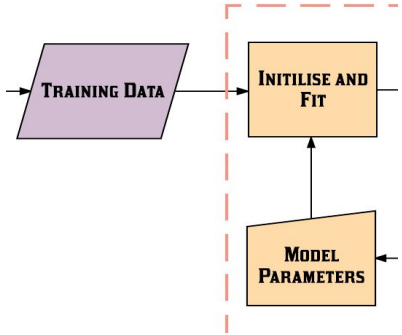


Image from: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

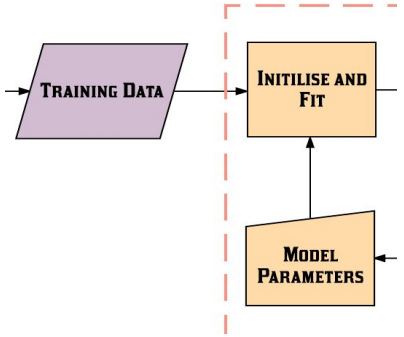
Basic ML Tutorial: 3. Train some ML models

Now we can train a couple of models to determine if someone is "Charged Off" on their loan. Here's a list of the more common ML models: note there are lots more!

1. K nearest-Neighbours (KNN) - sklearn.neighbors.KNeighborsClassifier
2. Support Vector Machines (SVM) - sklearn.svm.SVC or sklearn.svm.LinearSVC
3. Random Forests - sklearn.ensemble.RandomForestClassifier
4. Gradient Boosted Decision Trees - sklearn.ensemble.GradientBoostingClassifier
5. Logistic Regression - sklearn.linear_model.LogisticRegression



Basic ML Tutorial: 3. Train some ML models



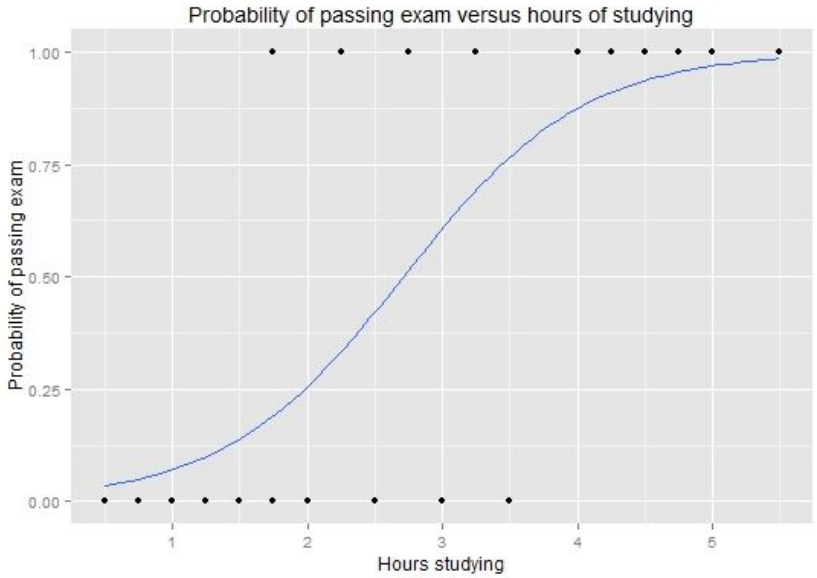
You can use anyone I'll use Logistic Regression.

```
>from sklearn.linear_model import LogisticRegression
>model = LogisticRegression()
>model.fit(X_train,y_train)
```

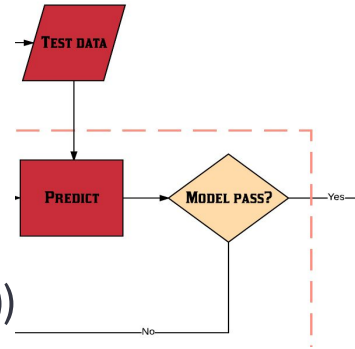
Yes that is it! Well not really ...

Logistic Regression

- Like linear regression it uses the training data to find the optimal parameters of a function, i.e. θ below
- Sigmoid Function: $h(\theta) = 1/(1+e^{-z})$ where $Z = \theta^T X$
- Minimizes the Cost function, J(θ) & θ
- Uses the derived θ-values to assign a probability to new data



Basic ML Tutorial: 4. Test your ML model!



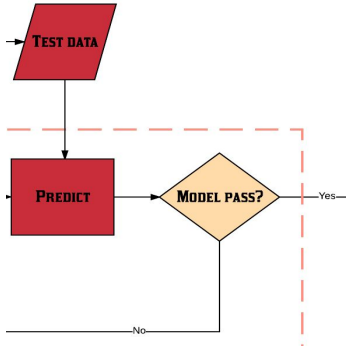

```
>y_pred_LR = model.predict(X_test)
>print('Accuracy of the model = {:.4f}'.format(model.score(X_test,y_test)))
Accuracy of the model = 0.9987
```

- Oh cool the model is 99.9% accurate! Well what does that mean? Is this really reliable?

To really determine the accuracy or the predictive power of your model, there are a few more useful tools than “Accuracy”

- Confusion matrix
- Classification report
- ROC curve (Precision and Recall curve for unbalanced datasets)
- Area Under the Curve(AUC)

Basic ML Tutorial: 5. Does the model pass?

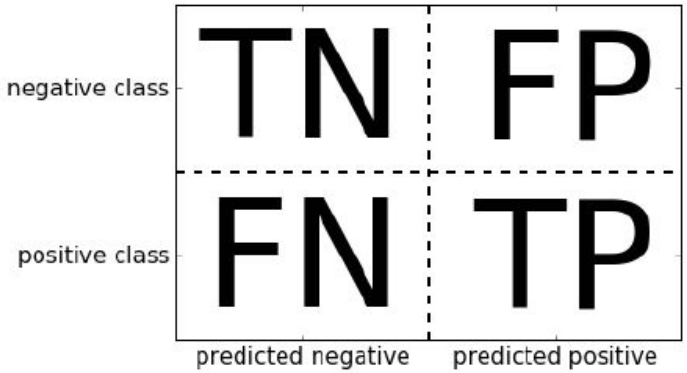


Confusion matrix

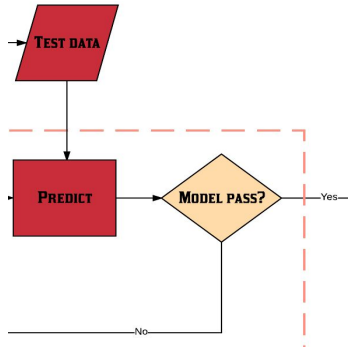
>from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve, average_precision_score

> print('confusion matrix:\n {}'.format(confusion_matrix(y_test,y_pred_LR)))

confusion matrix:
[[43777 1]
[67 8541]]



Basic ML Tutorial: 5. Does the model pass?



Classification report

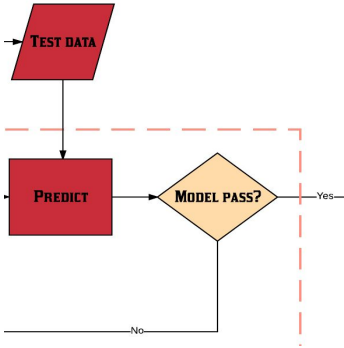
> print('Classification report: \n {}'.format(classification_report(y_test,y_pred_LR)))

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     43778
           1       1.00      0.99      1.00      8608

avg / total       1.00      1.00      1.00     52386
```

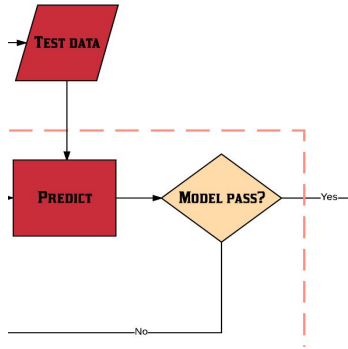
Basic ML Tutorial: 5. Does the model pass?

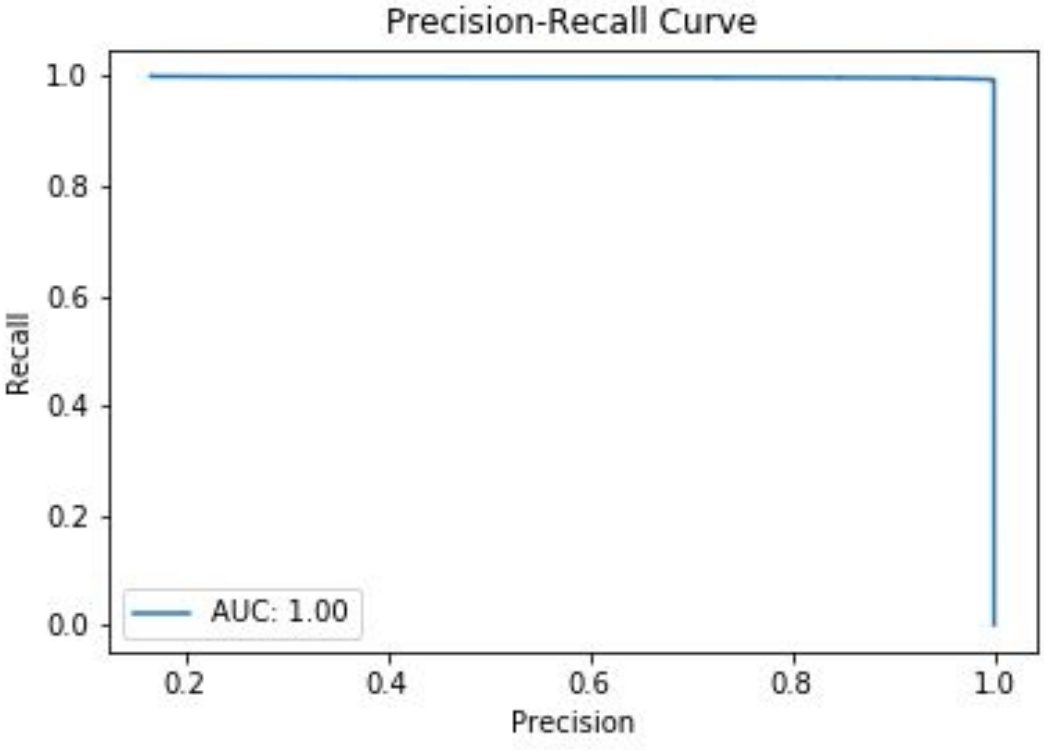


Precision-Recall Curve and AUC

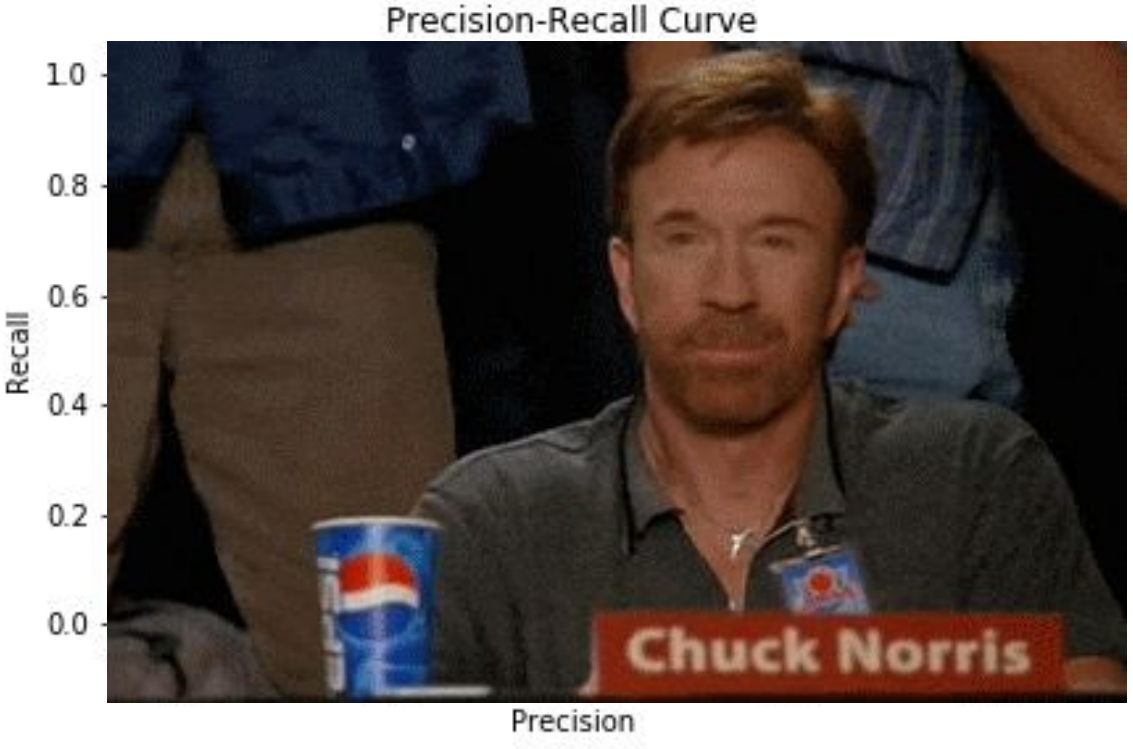
```
>precision, recall, thresholds =
precision_recall_curve(y_test,model.predict_proba(X_test)[:, 1])
>AUC = average_precision_score(y_test, model.predict_proba(X_test)[:, 1])
>import matplotlib.pyplot as plt
>plt.plot(precision, recall, label='AUC: {:.2f}'.format(AUC))
>plt.ylabel('Recall')
>plt.xlabel('Precision')
>plt.title('Precision-Recall Curve')
>plt.legend(loc='best')
```

Basic ML Tutorial: 5. Does the model pass?





Basic ML Tutorial: 5. Does the model pass?



What next?

If you are interested here are some invaluable resources:

- Textbooks:
  - Introduction to Machine Learning with python: https://www.dropbox.com/s/9e40qzxpw8dclu3/Introduction_to_Machine_Learning_with_Python.pdf?dl=0
  - An Introduction to Statistical Learning with Applications in R: http://www-bcf.usc.edu/~gareth/ISL/
  - Deep Learning: http://www.deeplearningbook.org/

What next?

If you are interested here are some invaluable resources:

Courses:

- Andrew Ng's Coursera course (good introduction, but he dives deep into the actual ML codes)
- Udemy tons of great courses and usually have sales for £10 per course
  - I learned from Python for Data Science and Machine Learning Bootcamp by Jose Portilla
- https://www.dataquest.io/ - a subscription fee if you want the advanced ML courses
- Deep Learning: https://machinelearningmastery.com/ (14 day free email course on Keras and Tflow) - really great introduction
- https://www.analyticsvidhya.com/trainings/

What next?

If you are interested here are some invaluable resources:

Data:

- Kaggle: https://www.kaggle.com/
- OpenML: https://www.openml.org/search?type=data
- UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php
- https://data.gov.uk/
- The internet!

What next?

If you are interested here are some invaluable resources:

ML Competitions:

- https://www.kaggle.com/competitions
- https://www.hackerearth.com/challenges/

JBCA Hack Nights and ML Group

- First one was in August, kicked off by David Mulcahy and Myself
  - Sponsored by Prof. A. Scaife
- Second one in October. Philippa and myself
- Third one: December (7th or 14th)
- Start a ML Group in JBCA.
  - Meet once per week on a Monday
  - Get involved in Kaggle competitions
- Previous challenges:
  - Aug17: https://github.com/Davidmul/JBCA_HACKNIGHT
  - Oct17: https://github.com/hrampadarath/JBCA_HackNight_Oct17