Predicting Hotel Cancellations with Machine Learning

Michael Grogan

Machine Learning Consultant @ michael-grogan.com

Why are hotel cancellations a problem?

- Inefficient allocation of rooms and other resources
- Customers who would follow through with bookings cannot do so due to lack of capacity
- Indication that hotels are targeting their services to the wrong groups of customers

How does machine learning help solve this issue?

- Allows for identification of factors that could lead a customer to cancel
- Time series forecasts can provide insights as to fluctuations in cancellation frequency
- Offers hotel businesses the opportunity to rethink their target markets

Original Authors

- Antonio, Almeida, Nunes (2016): Using Data Science to Predict Hotel Booking Cancellations.

- This presentation will describe alternative machine learning models that I have conducted on these datasets.

- Notebooks and datasets available at: https://github.com/MGCodesandStats.

Data Architecture

- Designing a machine learning model is only one component of an ML project.
- Under what environment will the model be run? Cloud? Locally?
- What are the relative advantages and disadvantages of each?

Amazon SageMaker: Some Advantages

Easier to coordinate Python versions across users

Ability to modify computing resources as needed to run models

No need for upfront investment

Running and maintaining a data center becomes unnecessary

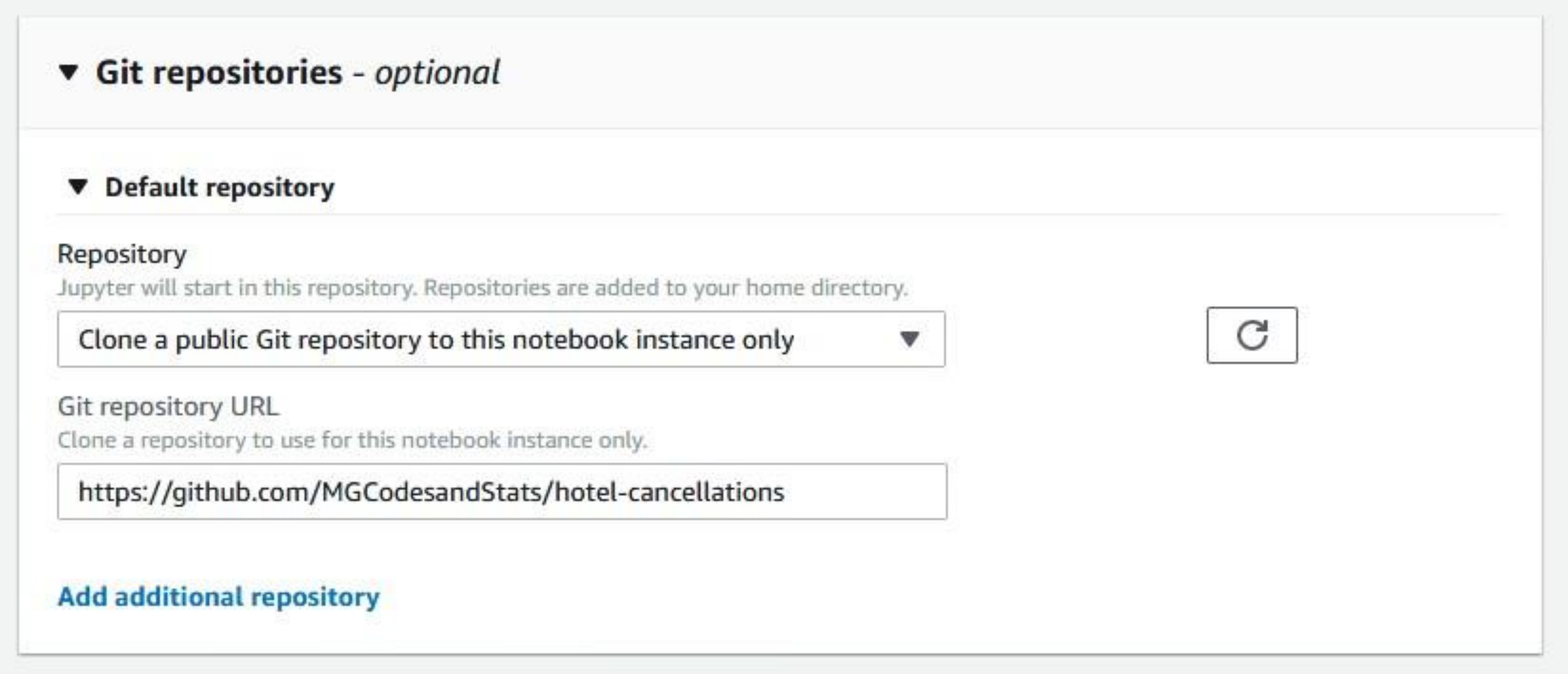
Sample workflow on Amazon SageMaker

Add repository from GitHub or AWS CodeCommit

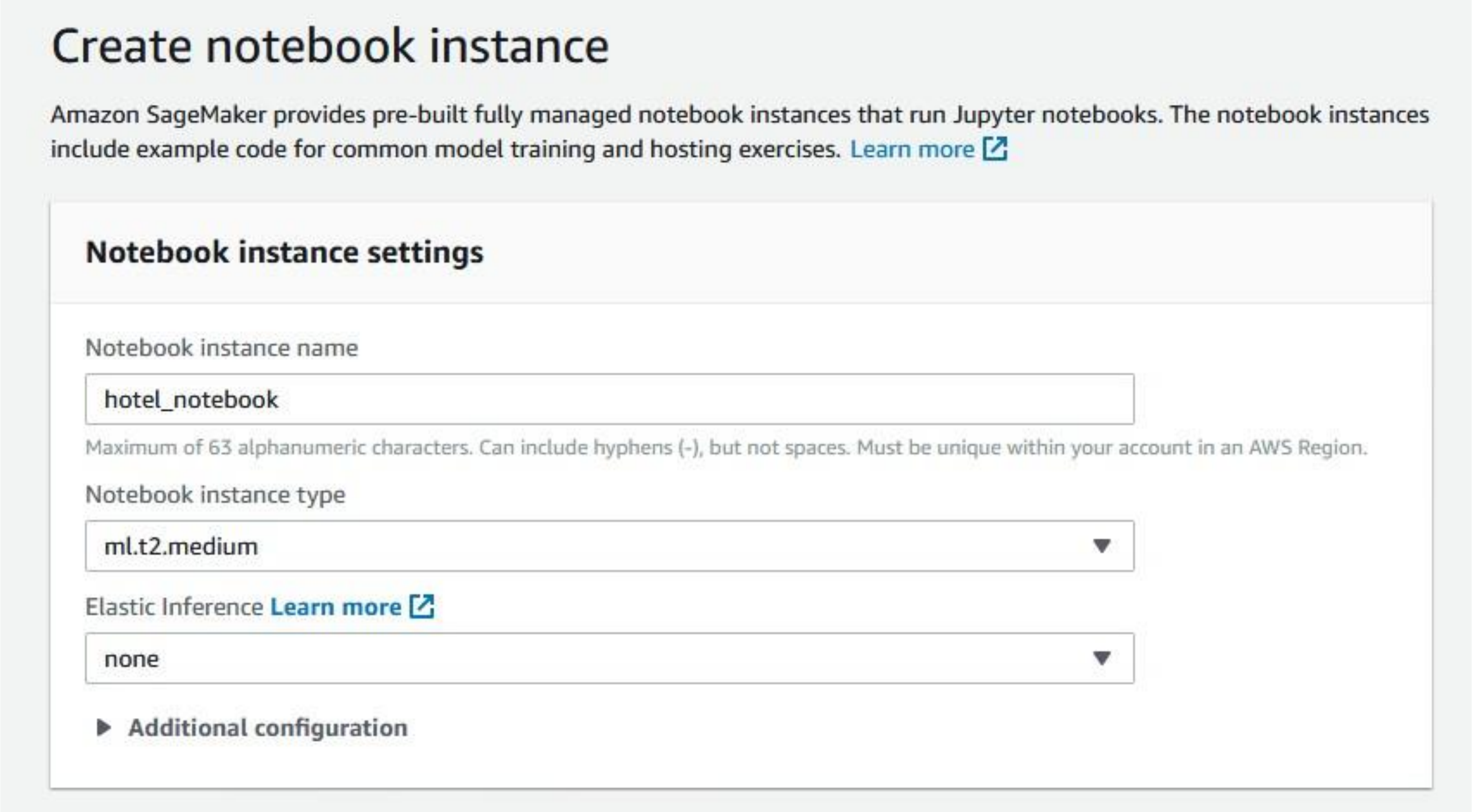
Select instance type, e.g. t2.medium, t2.large…

Create notebook instance and generate ML solution in the cloud

Add repository from GitHub or AWS CodeCommit



Select instance type, e.g. t2.medium, t2.large



Create notebook instance and generate ML solution in the cloud

```
In [23]: Y_train

Out[23]: array([0.25961538, 0.39423077, 0.26442308, 0.35576923, 0.64423077,
               0.29807692, 0.82692308, 0.52403846, 0.37019231, 0.88461538,
               0.00961538, 0.38461538, 0.14423077, 0.14903846, 0.19230769,
               0.23557692, 0.01923077, 0.54326923, 0.04807692, 0.11057692,
               0.3125    , 0.13942308, 0.10096154, 0.125     , 0.        ,
               0.        , 0.00480769, 0.20673077, 0.77884615, 0.09134615,
               0.57211538, 0.35576923, 0.16346154, 0.24519231, 0.5       ,
               0.24519231, 0.31730769, 0.40384615, 0.49038462, 1.        ,
               0.3125    , 0.62019231, 0.61057692, 0.32692308, 0.40865385,
               0.30288462, 0.46153846, 0.29326923, 0.31730769, 0.29326923,
               0.50480769, 0.50961538, 0.40384615, 0.52884615, 0.64903846,
               0.62980769, 0.5       , 0.44230769, 0.51442308, 0.25480769,
               0.55288462, 0.47115385, 0.5       , 0.34134615, 0.80769231,
               0.57692308, 0.46634615, 0.26923077, 0.12019231, 0.21634615,
               0.28846154, 0.20673077, 0.10576923, 0.33653846])

In [24]: # reshape input to be [samples, time steps, features]
         X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
         X_val = np.reshape(X_val, (X_val.shape[0], 1, X_val.shape[1]))

         # Generate LSTM network
         model = tf.keras.Sequential()
         model.add(LSTM(4, input_shape=(1, previous)))
         model.add(Dense(1))
         model.compile(loss='mean_squared_error', optimizer='adam')
         model.fit(X_train, Y_train, epochs=20, batch_size=1, verbose=2)
```

Pricing Samples: On-Demand ML Notebook Instances (Standard)

US East (N. Virginia)

Building

On-Demand ML Notebook Instances

Standard Instances - Current Generation	Price per Hour
ml.t2.medium	$0.0464
ml.t2.large	$0.1299
ml.t2.xlarge	$0.2598
ml.t2.2xlarge	$0.5197
ml.t3.medium	$0.0582
ml.t3.large	$0.1165
ml.t3.xlarge	$0.233

Pricing Samples: On-Demand ML Notebook Instances (Accelerated Computing)

Accelerated Computing - Current Generation	
ml.p3.2xlarge	$4.284
ml.p3.8xlarge	$17.136
ml.p3.16xlarge	$34.272
ml.g4dn.xlarge	$0.736
ml.g4dn.2xlarge	$1.053
ml.g4dn.4xlarge	$1.686
ml.g4dn.8xlarge	$3.046
ml.g4dn.12xlarge	$5.477
ml.g4dn.16xlarge	$6.093

Three components

- Identifying important customer features
  - ExtraTreesClassifier
- Classifying potential customers in terms of cancellation risk
  - SVM, XGBoost
- Forecasting fluctuations in hotel cancellation frequency
  - ARIMA, LSTM

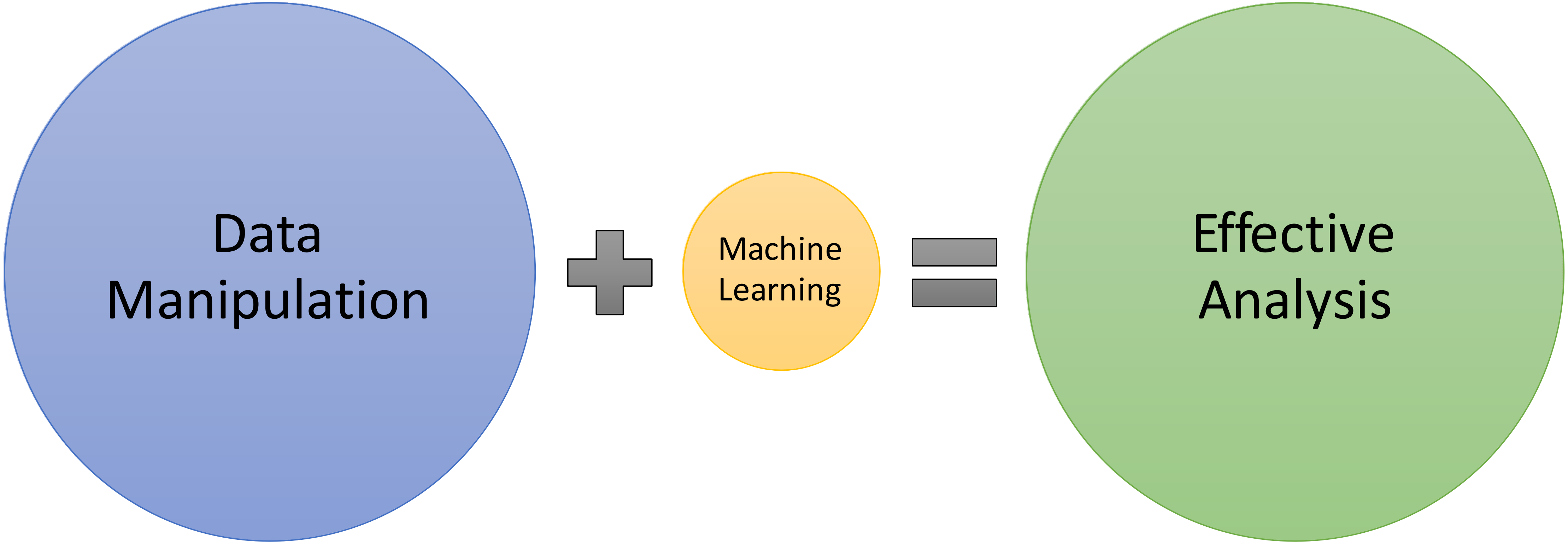
Question

What do you think is the most important Python library in a machine learning project?

Answer

pandas

Most of the machine learning process… is not machine learning



You may have data – but it is not the data you want

What we have is a classification set:

IsCanceled	LeadTime	ArrivalDateYear	ArrivalDateMonth	ArrivalDateWeekNumber	ArrivalDateDayOfMonth
0	6	0	5	26	0
1	88	0	5	26	0
1	65	0	5	26	0
1	92	0	5	26	0
1	100	0	5	26	1
1	79	0	5	26	1
0	3	0	5	26	1
1	63	0	5	26	1
1	62	0	5	26	1
1	62	0	5	26	1
0	43	0	5	26	2

What we want is a time series:

201527	41.0
201528	48.0
201529	87.0
201530	74.0
201531	101.0
201532	68.0
201533	96.0
201534	69.0
201535	88.0
201536	148.0
201537	76.0
201538	186.0
201539	123.0

Data Manipulation with pandas

1. Merge year and week number

```
In [6]: from pandas import DataFrame

        df = DataFrame(c, columns= ['ArrivalDateYear', 'ArrivalDateWeekNumber'])
        df
```

Out[6]:

	ArrivalDateYear	ArrivalDateWeekNumber
0	2015	27
1	2015	27
2	2015	27
3	2015	27
4	2015	27
5	2015	27

```
df1 = df['ArrivalDateYear'].map(str) + df['ArrivalDateWeekNumber'].map(str)
print (df1)
df1=pd.DataFrame(df1)
```

```
0        201527
1        201527
2        201527
3        201527
4        201527
5        201527
6        201527
7        201527
```

Data Manipulation with pandas

2. Merge dates and cancellation incidences

In [10]:
```
df3=pd.concat([df1, df2], axis = 1)
df3
df3.columns = ['FullDate', 'IsCanceled']
```

In [11]:
```
df3
df3.sort_values(['FullDate','IsCanceled'], ascending=True)
```

Out[11]:

	FullDate	IsCanceled
0	201527	0.0
6	201527	0.0
10	201527	0.0
11	201527	0.0
12	201527	0.0
13	201527	0.0
15	201527	0.0
17	201527	0.0

Data Manipulation with pandas

3. Sum weekly cancellations and order by date

In [12]:

```
df4 = df3.groupby('FullDate').agg(sum)
df4
df4.sort_values(['FullDate'], ascending=True)
```

Out[12]:

	IsCanceled
FullDate	
201527	97.0
201528	153.0
201529	228.0
201530	321.0
201531	159.0
201532	308.0
201533	428.0
201534	191.0
201535	212.0

Feature Selection – What Is Important?

- Of all the potential features, only a select few are important in classifying future bookings in terms of cancellation risk.

- ExtraTreesClassifier is used to rank features – the higher the score, the more important the feature – in most cases…

15	0.013381
20	0.017885
19	0.018724
9	0.033227
17	0.033918
1	0.040703
8	0.059414
21	0.061054
23	0.636264

Feature Selection – What Is Important?

- Top six features as identified by **ExtraTreesClassifier** and **Forward and Backward Feature Selection** Techniques:

- Lead time
- Country of origin
- Market segment
- Deposit type
- Customer type
- Required car parking spaces
- Arrival Date: Year
- Arrival Date: Month
- Arrival Date: Week Number
- Arrival Date: Day of Month

- Note that while Reservation Status was identified as the highest scoring feature, it is not included due to lack of theoretical validity - i.e. the variable represents cancellations vs. non-cancellations which has the same meaning as the response variable. Including this feature will negatively skew model results.

Accuracy

90% is great. 100% means you've overlooked something.

Training accuracy

Accuracy of the model in predicting other values in the training set (the dataset which was used to train the model in the first instance).

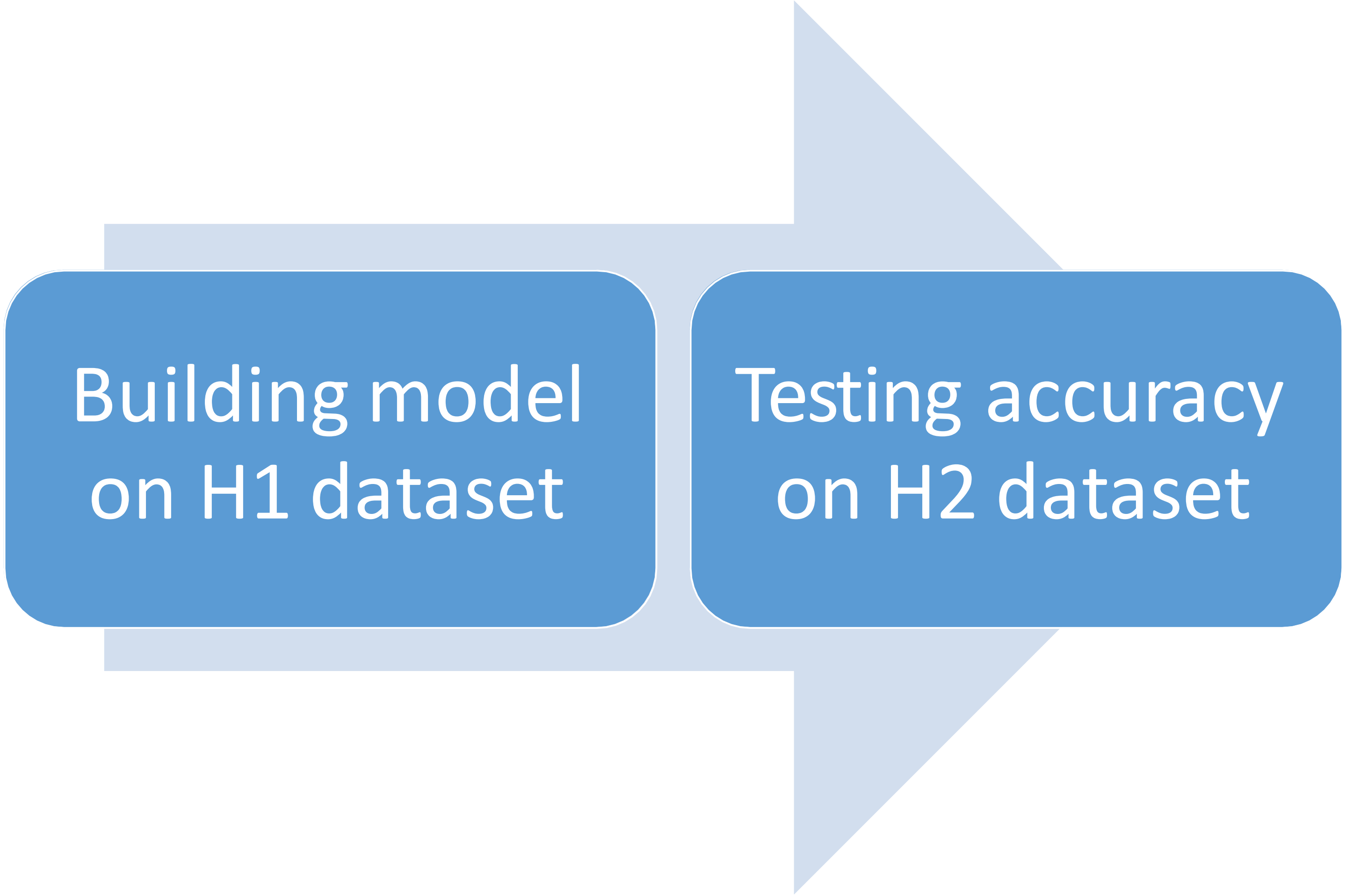
Validation accuracy

Accuracy of the model in predicting a segment of the dataset which has been "split off" from the training set.

Test accuracy

Accuracy of the model in predicting completely unseen data. This metric is typically seen as the litmus test to ensure a model's predictions are reliable.

Classification: Support Vector Machines



Classification Accuracy: Precision vs. Recall

Precision

((True Positive)/(True Positive + False Positive))

Recall

((True Positive)/(True Positive + False Negative))

SVM performance on test set

```
In [22]: from sklearn.metrics import classification_report,confusion_matrix
         print(confusion_matrix(b,prh2))
         print(classification_report(b,prh2))

[[25217 21011]
 [ 8436 24666]]
              precision    recall  f1-score   support

           0       0.75      0.55      0.63     46228
           1       0.54      0.75      0.63     33102

    accuracy                           0.63     79330
   macro avg       0.64      0.65      0.63     79330
weighted avg       0.66      0.63      0.63     79330
```

XGBoost performance on test set

scale_pos_weight = 3

```
In [21]: from sklearn.metrics import classification_report,confusion_matrix
         print(confusion_matrix(b,prh2))
         print(classification_report(b,prh2))
```

```
[[12650 33578]
 [ 1972 31130]]
              precision    recall  f1-score   support

           0       0.87      0.27      0.42     46228
           1       0.48      0.94      0.64     33102

    accuracy                           0.55     79330
   macro avg       0.67      0.61      0.53     79330
weighted avg       0.70      0.55      0.51     79330
```

Building a regression neural network

Input layer

Number of features in the training set + 1

10 features + 1 = 11 input neurons

Building a regression neural network

Hidden layer

Training Data Samples/
(Factor * (Input Neurons + Output Neurons))

24036/(1 * (11 + 1)) = 2003

Building a regression neural network

Output layer

`One neuron included in output layer by default`

Tradeoff: Number of iterations vs. batch size

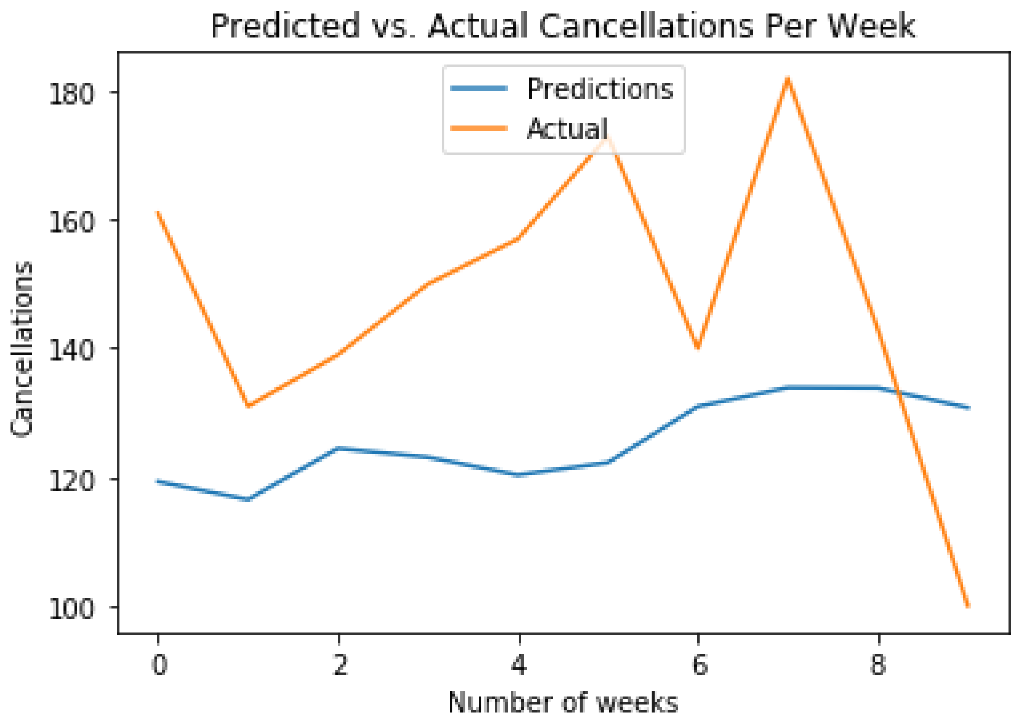
- **Batch size:** Refers to the number of training examples per one forward/backward pass.

- **Epoch:** One forward and backward pass for all training examples.

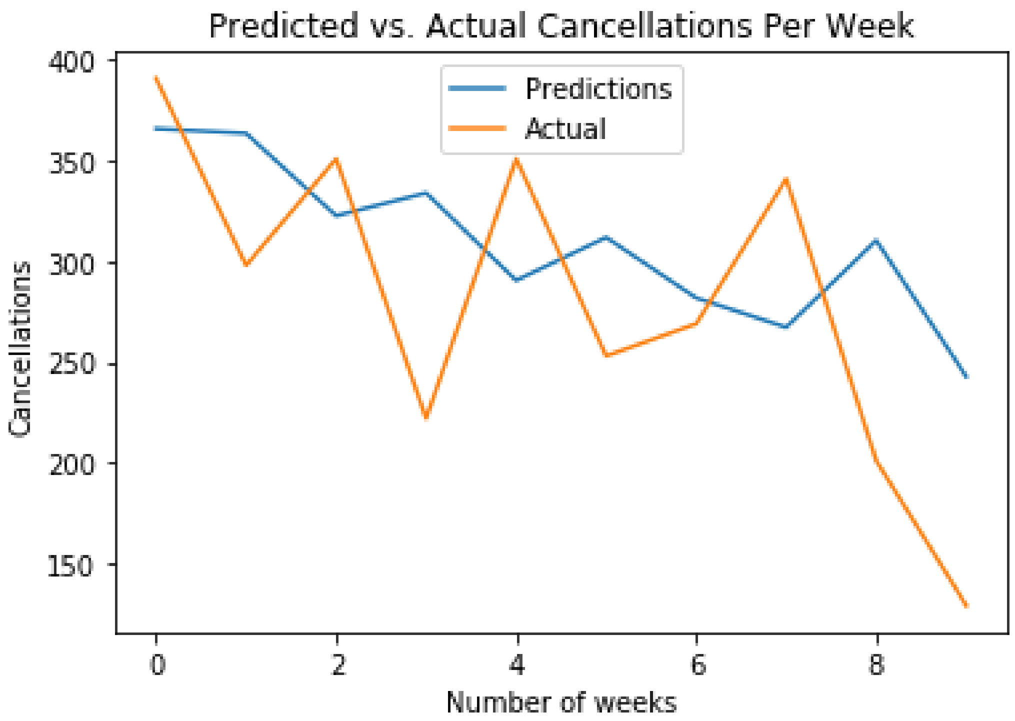
For instance, the training data contains 24,036 samples and the batch size is 150. This means that **160** iterations are required to complete 1 epoch.

Two time series – what is the difference?

H1

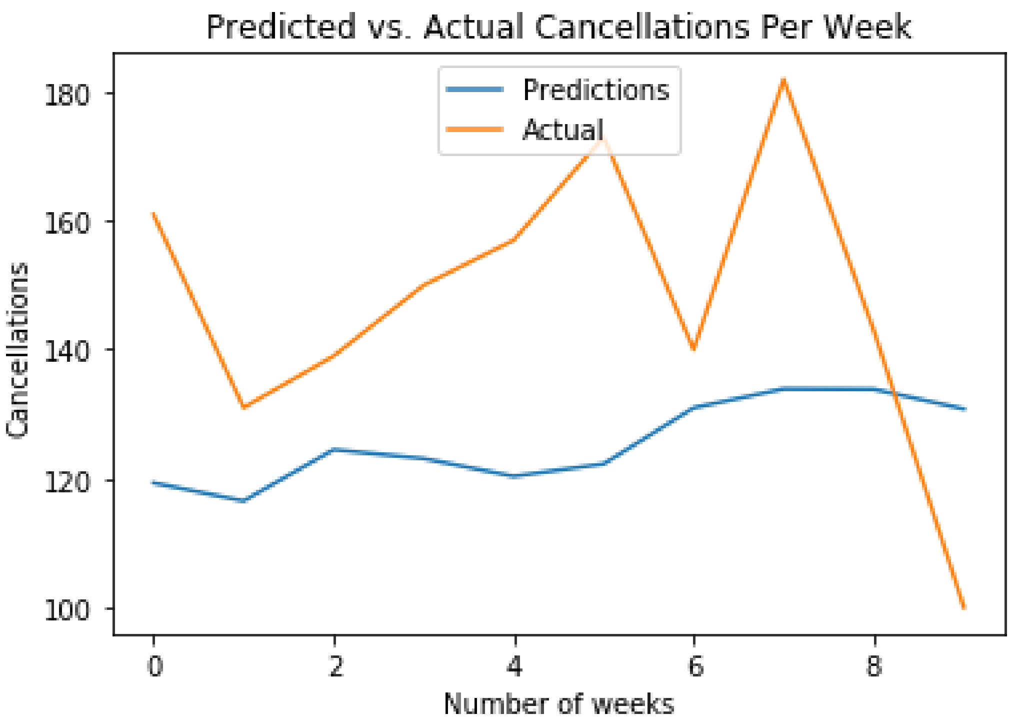


H2



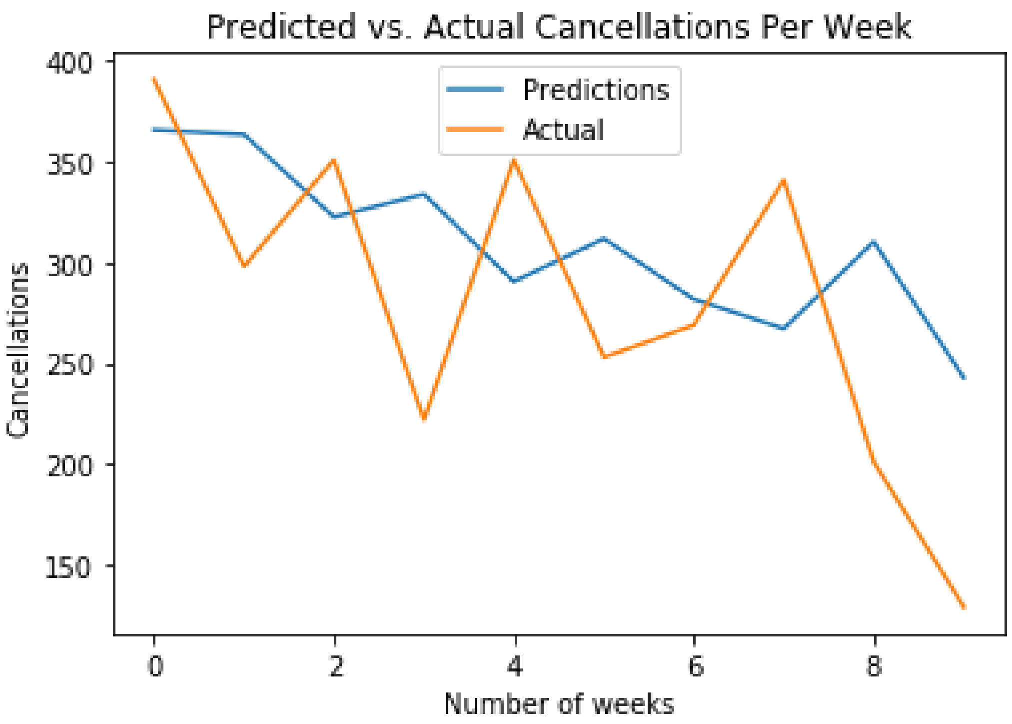
Findings

H1



ARIMA performed better

H2



LSTM performed better

ARIMA

Major tool used in time series analysis to attempt to forecast future values of a variable based on its present value.

- **p** = number of autoregressive terms
- **d** = differences to make series stationary
- **q** = moving average terms (or lags of the forecast errors)

LSTM (Long-Short Term Memory Network)

- Traditional neural networks are not particularly suitable for time series analysis.
- This is because neural networks do not account for the **sequential** (or step-wise) nature of time series.
- In this regard, a long-short term memory network (or **LSTM** model) must be used in order to examine long-term dependencies across the data.
- LSTMs are a type of **recurrent** neural network and work particularly well with volatile data.

Constructing an LSTM model

Choosing the time parameter

- In this case, the cancellation value at time *t* is being predicted by the previous **5** values

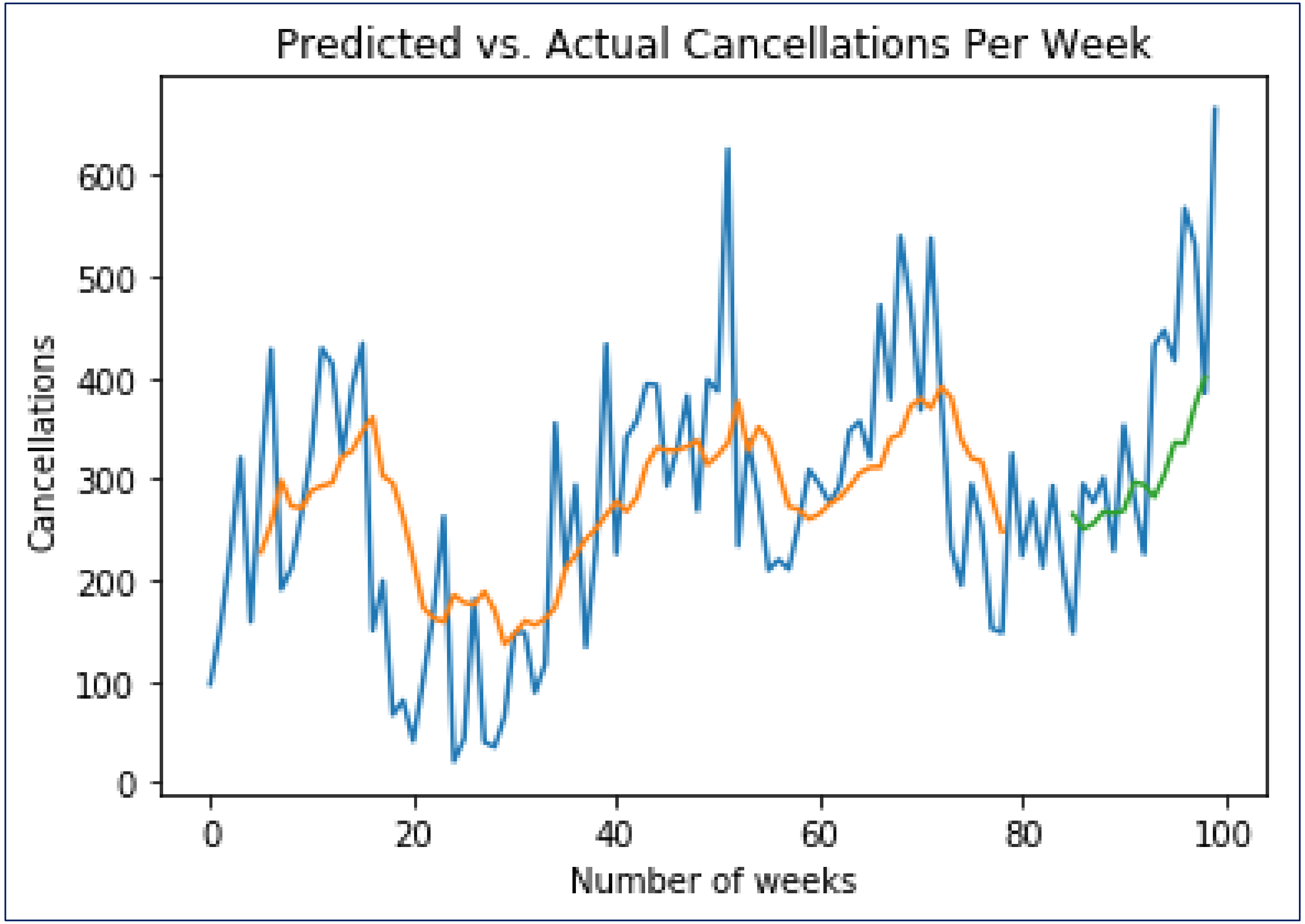
Scaling data appropriately

- *MinMaxScaler* used to scale data between 0 and 1

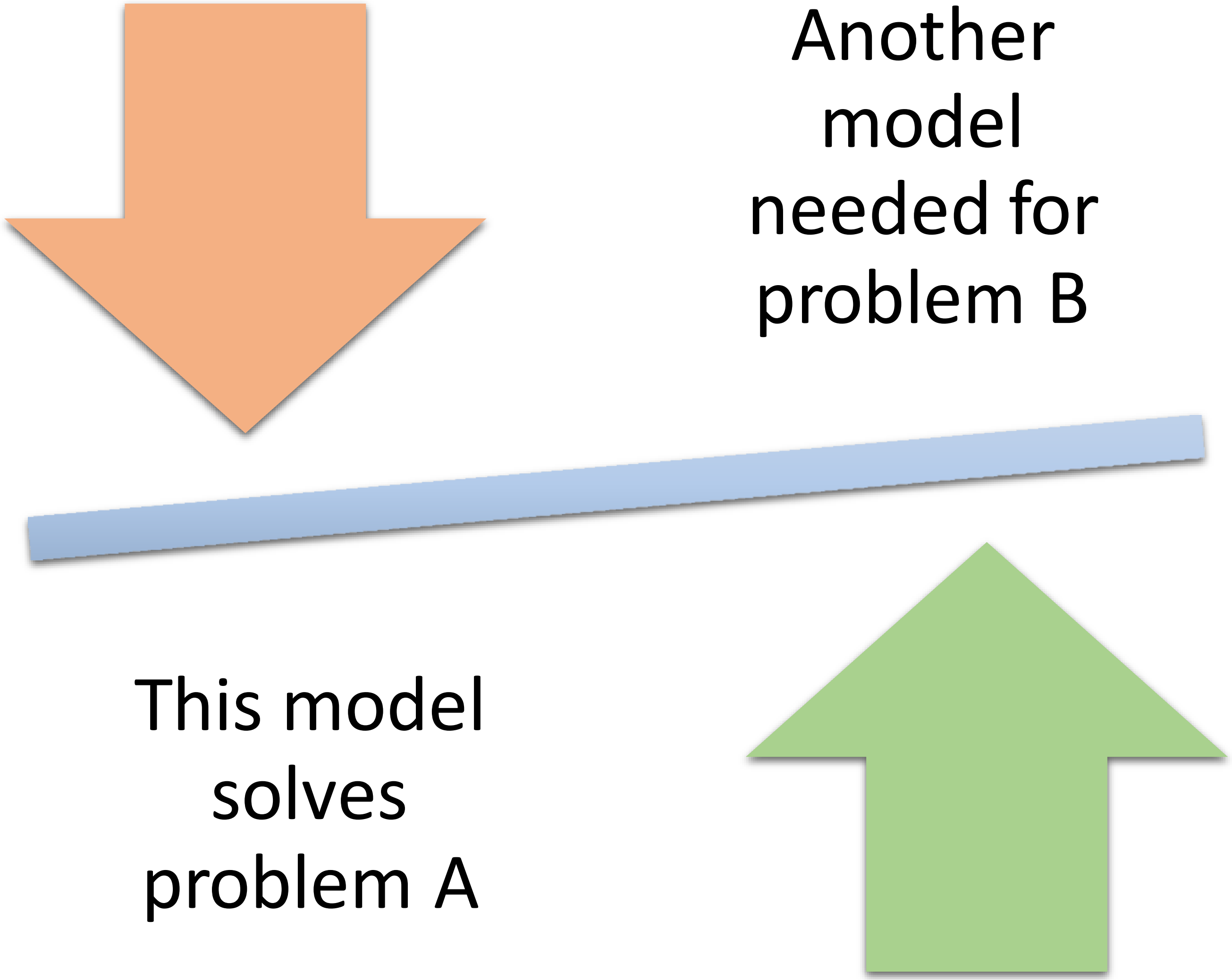
Configure neural network

- Loss = Mean Squared Error
- Optimizer = adam
- Trained across 20 epochs – further iterations proved redundant

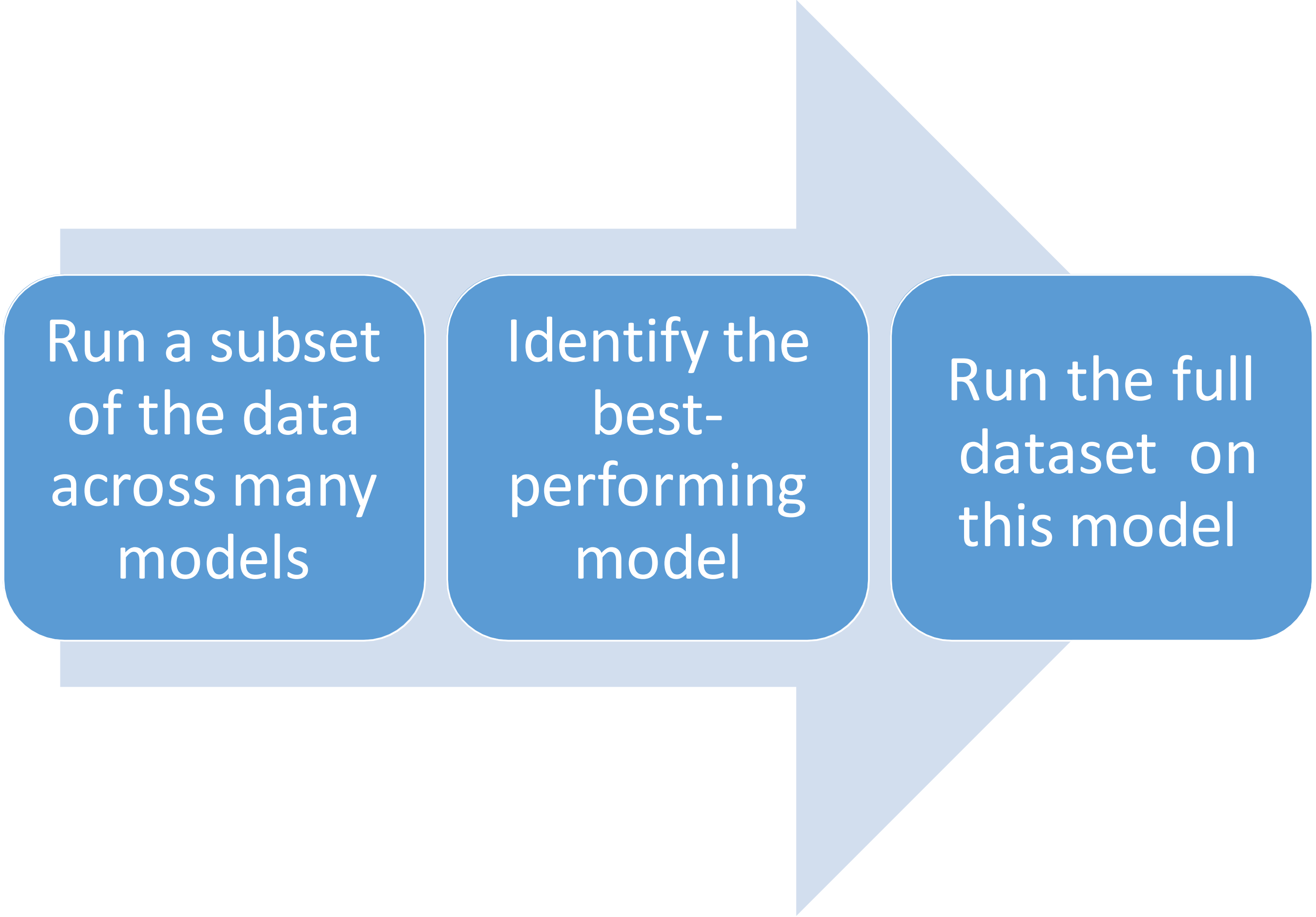
LSTM Results for H2 Dataset



“No Free Lunch” Theorem



Model Selection Considerations



Time Series Forecasting Accuracy Metrics

Metric	ARIMA	LSTM
MDA	0.86	0.8
RMSE	57.95	57.68
MFE	-12.72	-49.85

H1

Metric	ARIMA	LSTM
MDA	0.86	0.8
RMSE	274.08	112.58
MFE	156.33	48.94

H2

Conclusion

Data Manipulation is an integral part of an ML project

"No free lunch" – make sure the model is appropriate to the data

Pay attention to the workflow(s) being used and the relative advantages and disadvantages of each