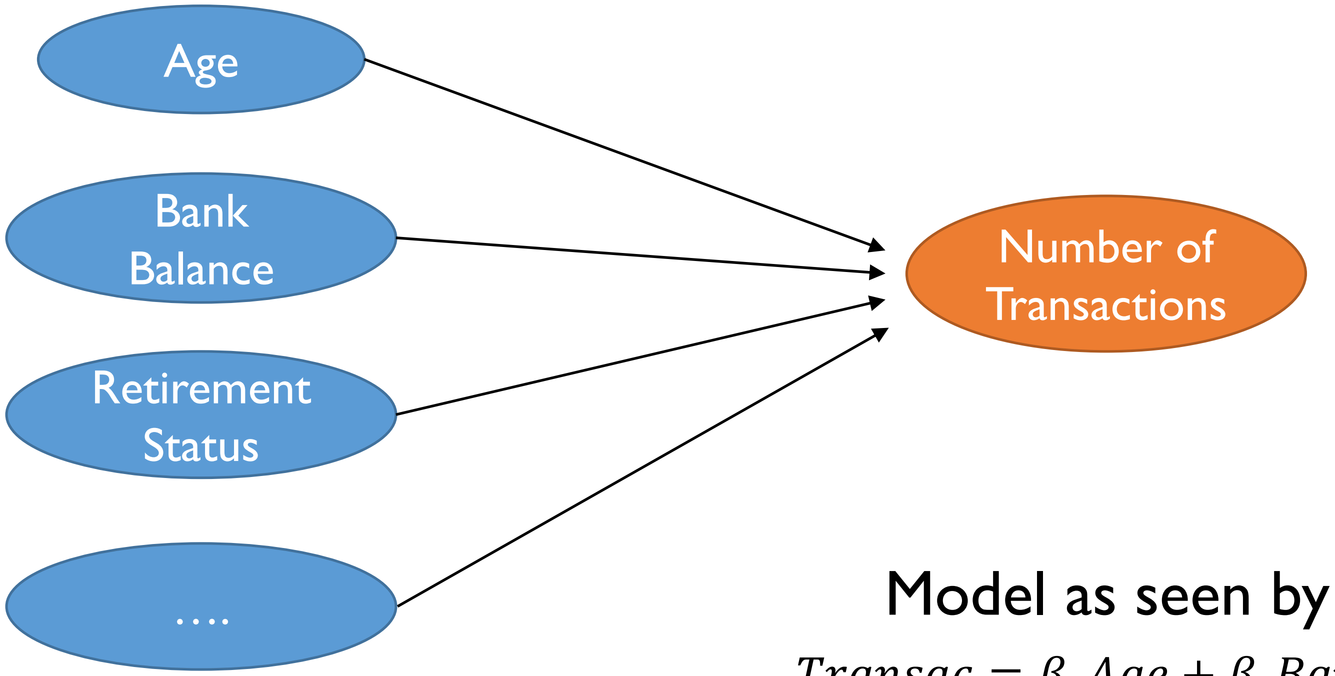
Introduction to deep learning

What is deep learning ?

- You need to predict how many transactions each customer will make next year.

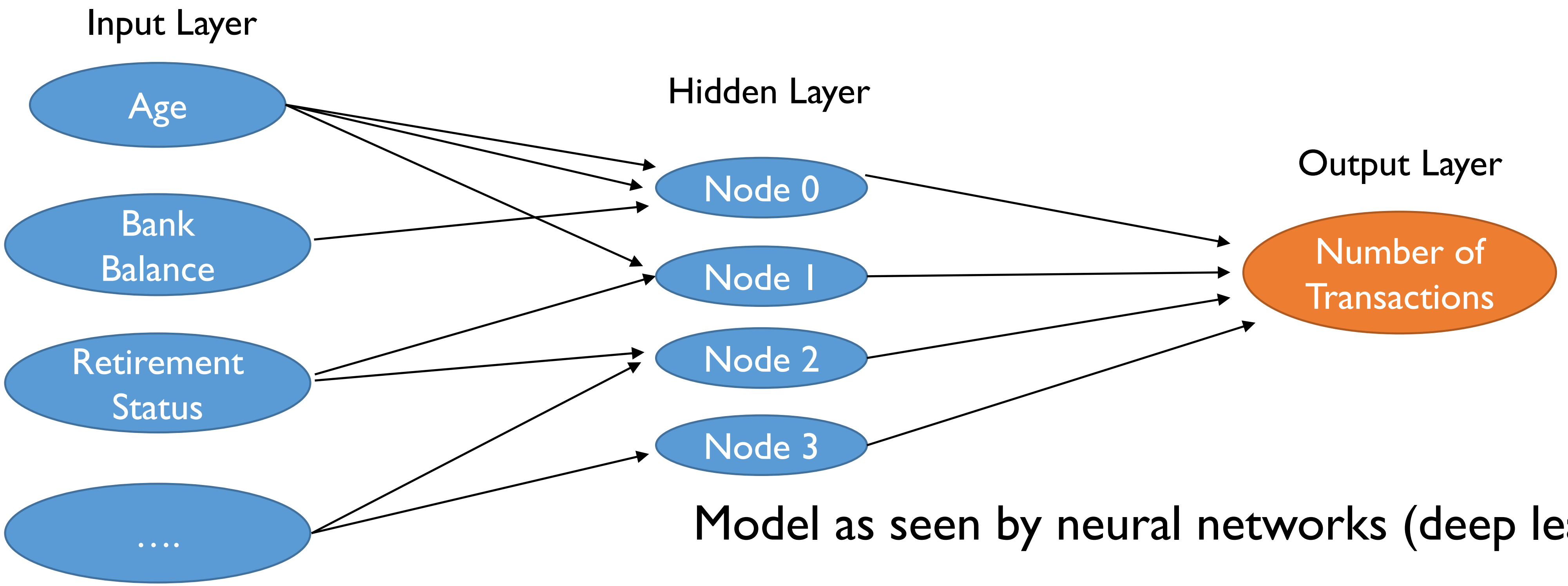


Model as seen by linear regression

$$Transac = \beta_0 Age + \beta_1 BankBalance + \beta_2\ Retired + \ldots$$

What is deep learning ?

- Deep learning is about neural networks and lots of computing power



Model as seen by neural networks (deep learning)

Why deep learning ?

- Extremely versatile, it works really well in lots of verticals
  - Text
  - Images
  - Videos
  - Audio
  - Time series
  - Panel data
- It was invented 20 years ago (neural networks), not enough computing power at the time.
- Refer to Annex for an overall introduction on how they work

Today you will learn

- How to provision computing services in the cloud
- How to develop and run python code for data science purposes
- How to develop and fit deep basic learning models

Agenda

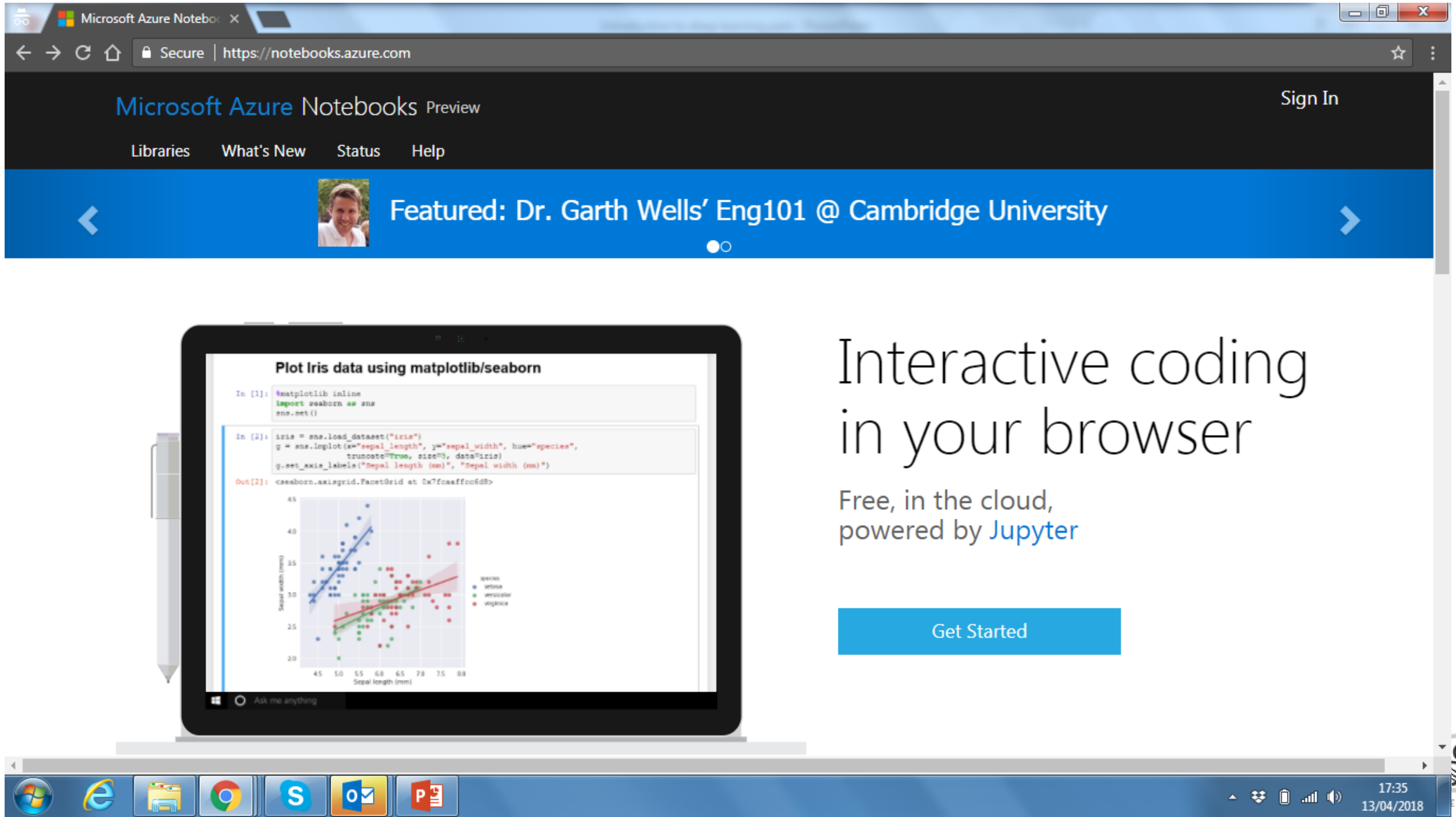
- Signing up for cloud computing services (15 minutes)
- Using azure notebooks (10 minutes)
- Deep Learning models: Regression example. (30 minutes)
- Deep Learning models: Classification example. (30 minutes)

Let’s practice

1. Signing up in cloud services

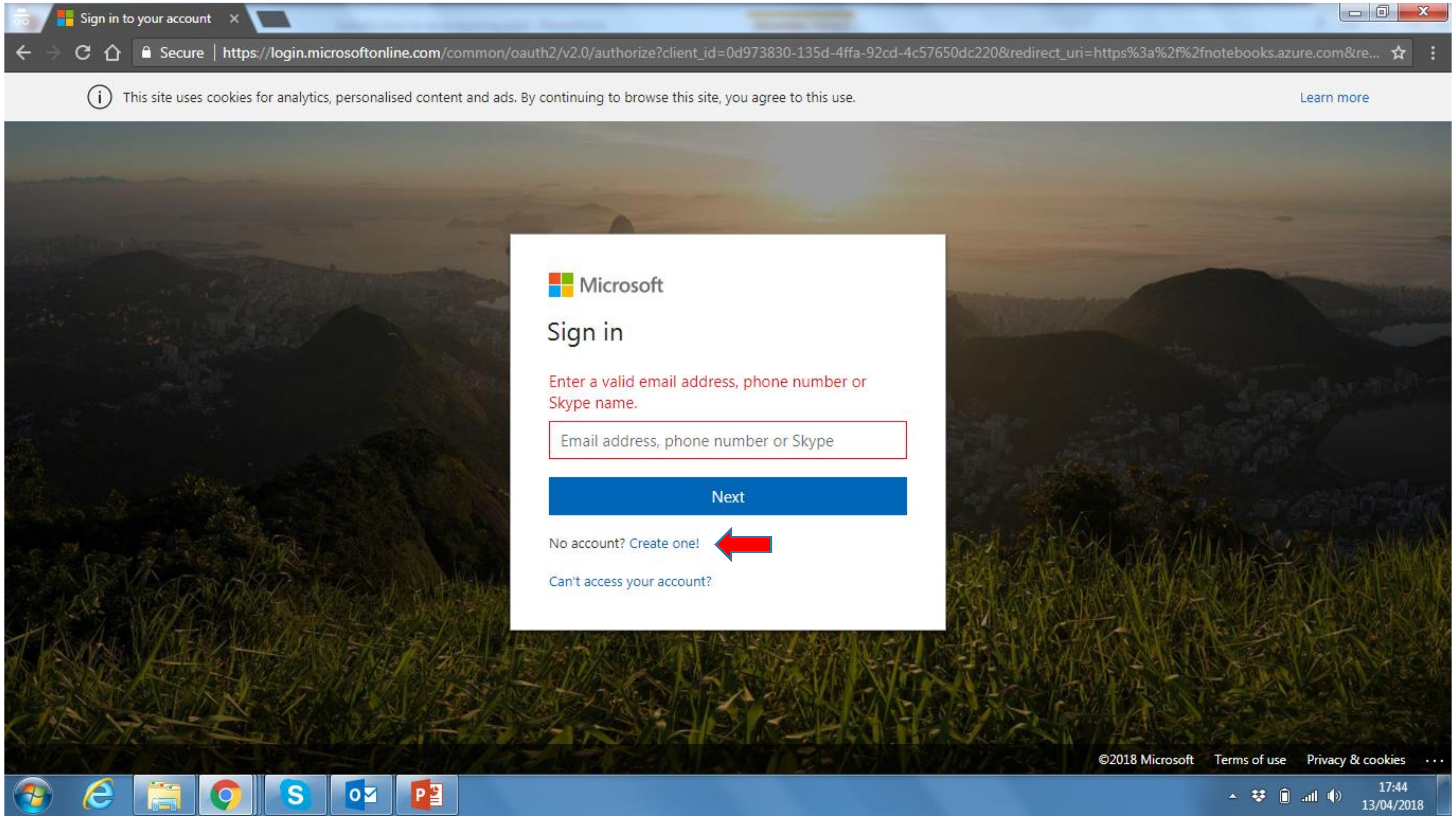
Microsoft Azure Notebooks

- You probably have an account already. Try to sign in at: https://notebooks.azure.com/ using your university/corporate account (e.g. dl0022@surrey.ac.uk)



Microsoft Azure Notebooks

- If not, please sign up using your personal email

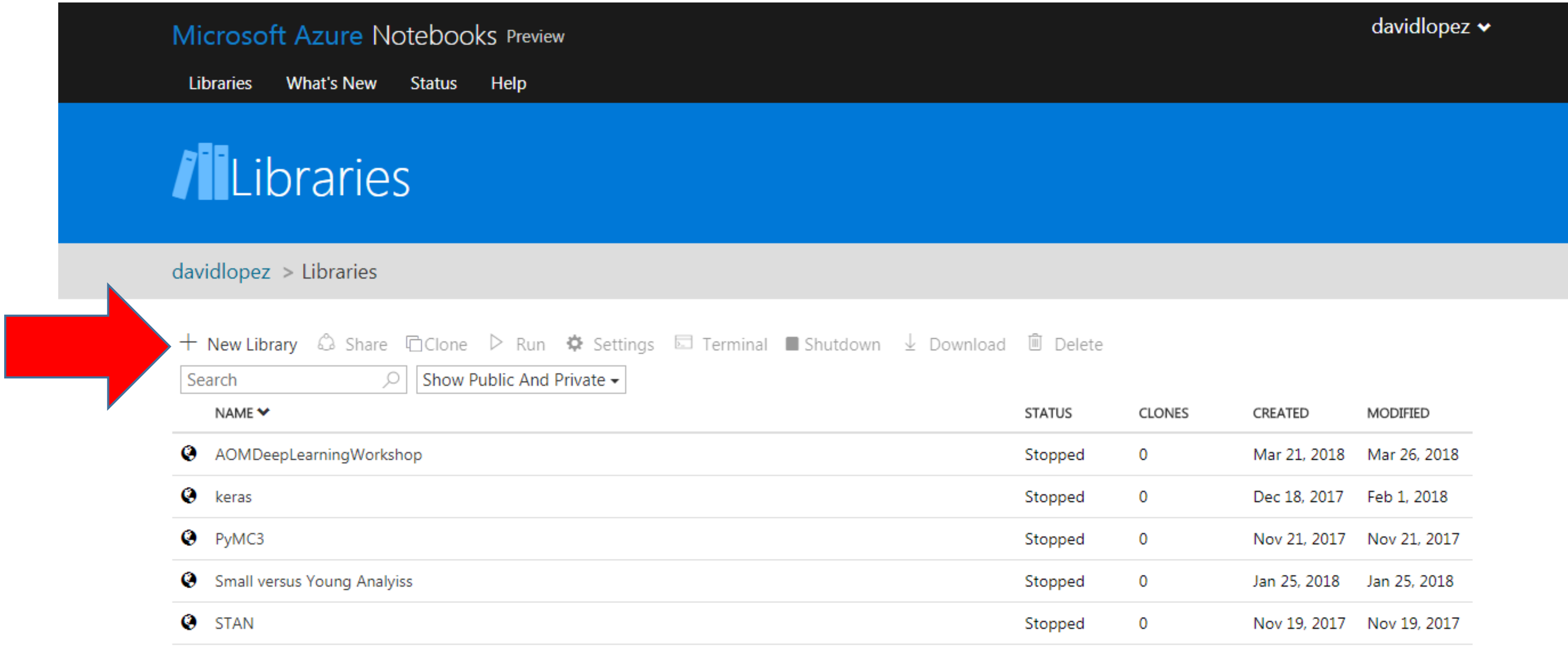


Let’s practice

II. Using Azure Notebooks

Microsoft Azure Notebooks

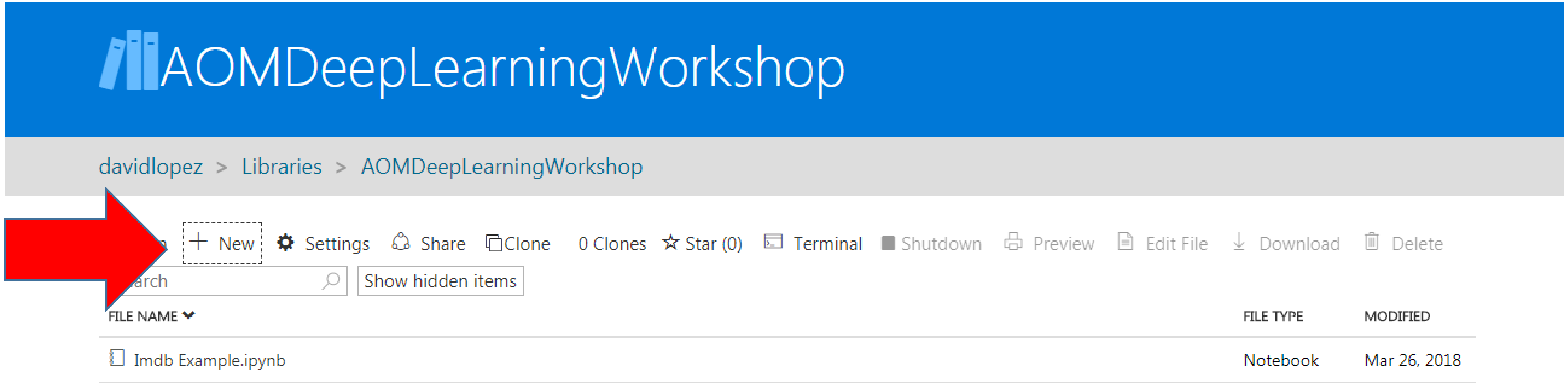
- You need to create a library in the first place. This is where your software code will live
- Click on "+ New Library". Give it a friendly name (e.g. AOMDeepLearningWorkshop) and a library ID (e.g. AOMDeepLearningWorkshop)



Imagine you work for a bank

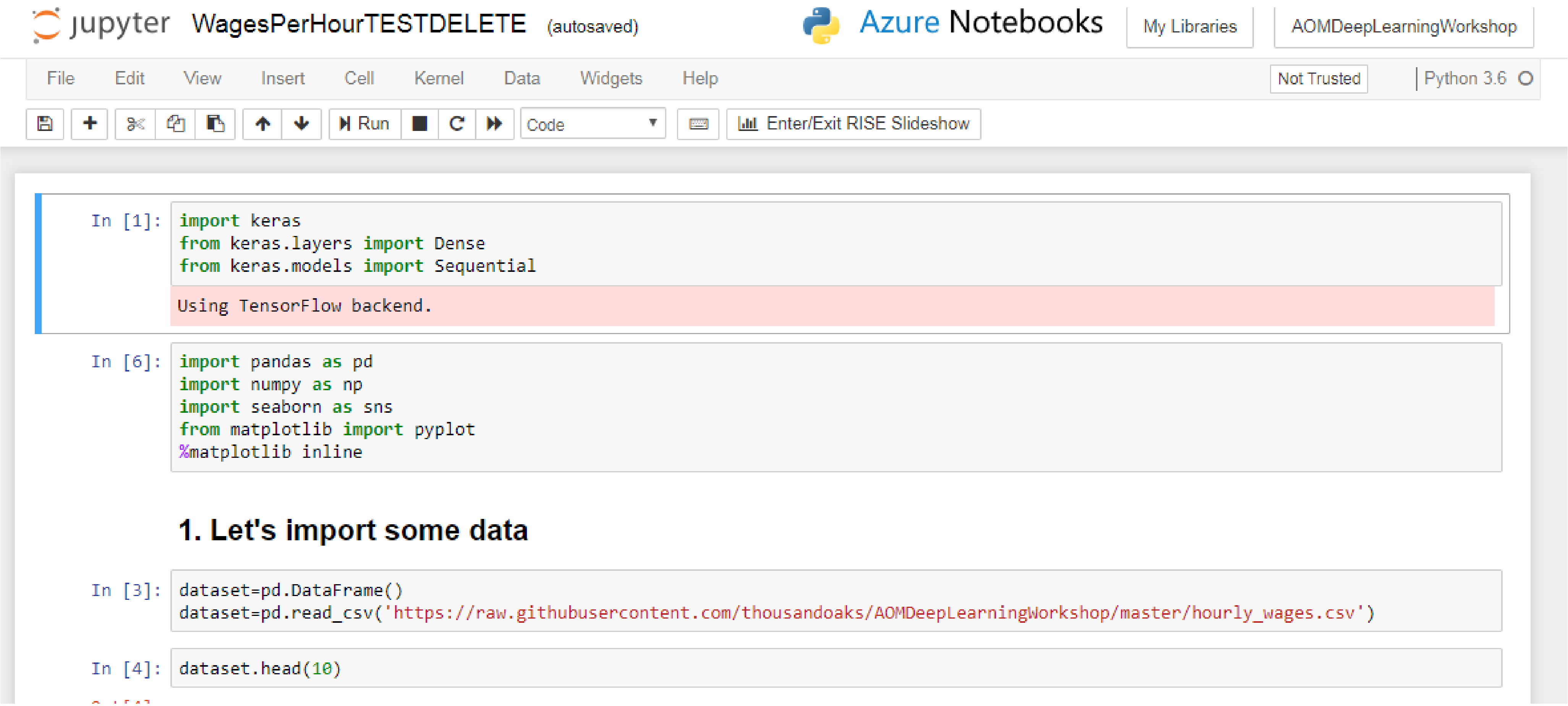
- Let's import the code which we will be using in the next exercise.
- Go to your Azure library (e.g. AOMDeepLearningWorkshop)
- Import a Notebook. Click on "+ New", then "From URL", insert the following URL:

https://github.com/thousandoaks/AOMDeepLearningWorkshop/blob/master/WagesPerHour.ipynb



Microsoft Azure Notebooks

- You are ready to rock !!

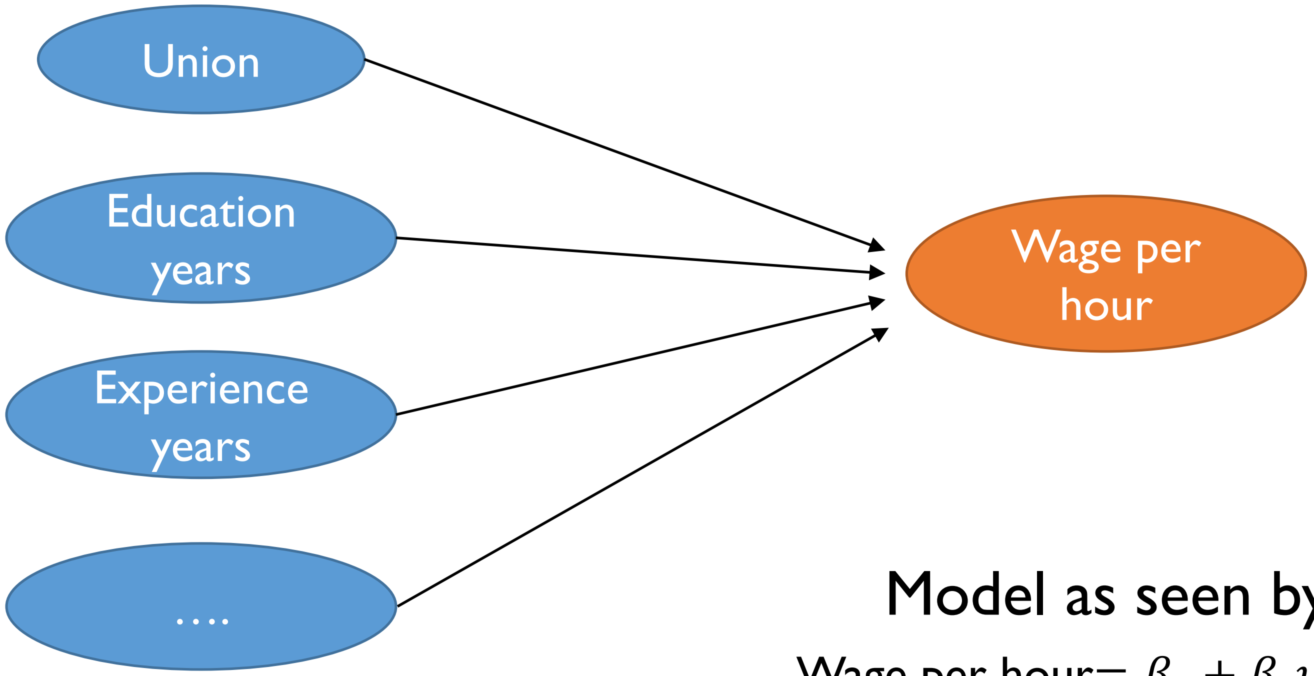


Let’s practice

III. Regression example.

Imagine you work for a bank

- You need to predict the hourly wage of customers based on several variables ("features" in AI parlance)

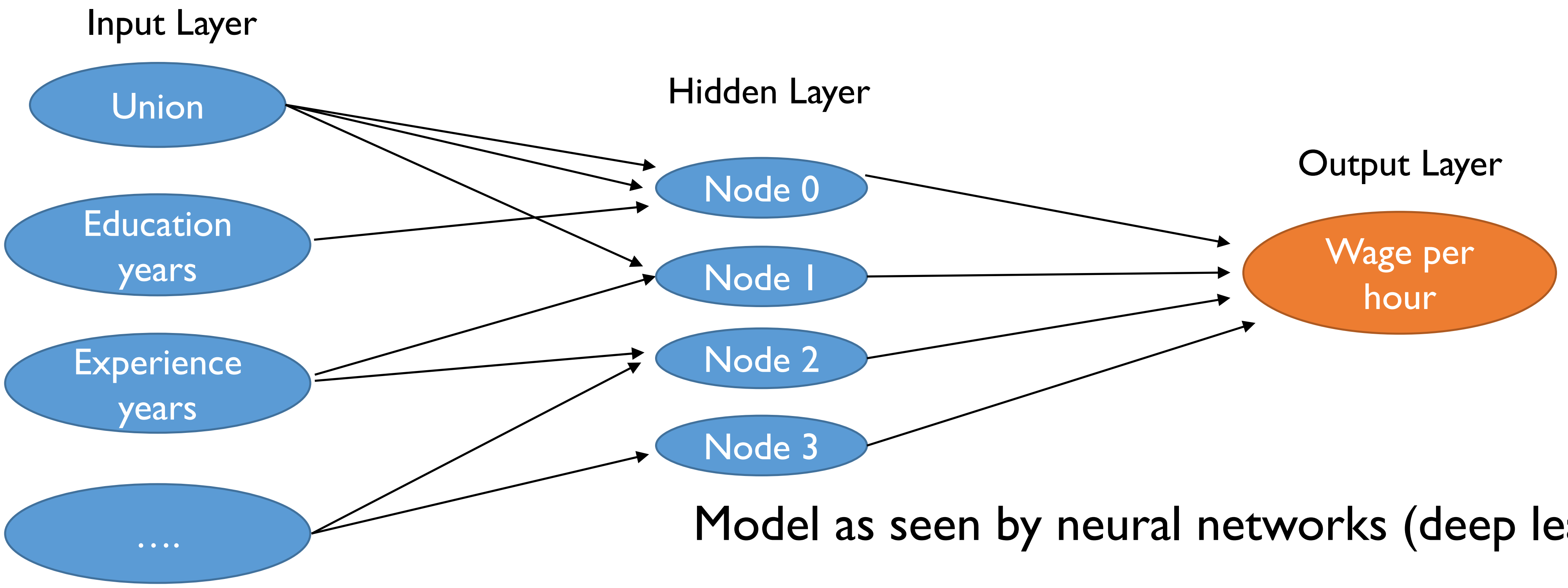


Model as seen by linear regression

Wage per hour= $\beta_0 + \beta_1 union + \beta_2\ educ\_years + \ldots$

Imagine you work for a bank

- You need to predict how many transactions each customer will make next year.



Model as seen by neural networks (deep learning)

Model building steps

A. Import the data
B. Understand the data (exploratory analysis, clustering, correlation)
C. Specify Architecture of the model
D. Compile the model
E. Fit the model
F. Make predictions

Model building steps

A. Import the data

```
In [3]: dataset=pd.DataFrame()
        dataset=pd.read_csv('https://raw.githubusercontent.com/thousandoaks/AOMDeepLearningWorkshop/master/hourly_wages.csv')

In [4]: dataset.head(10)
```

Out[4]:

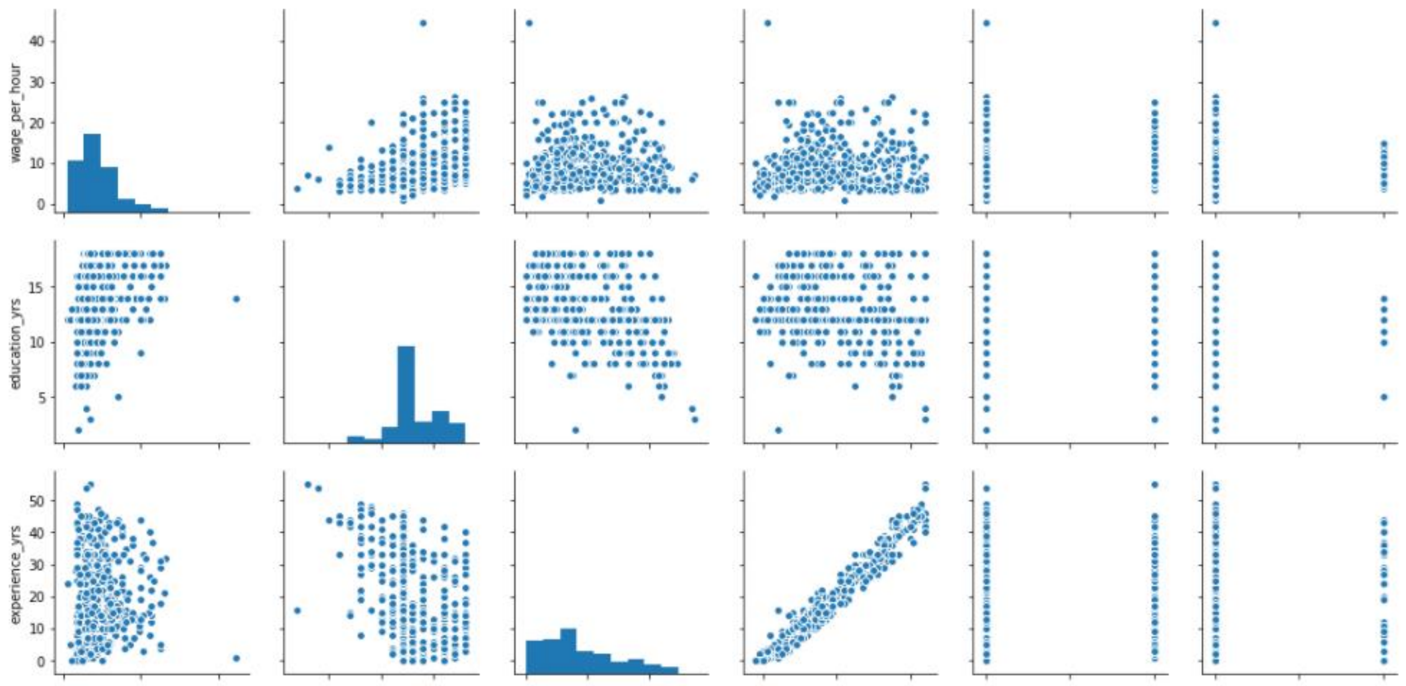
	wage_per_hour	union	education_yrs	experience_yrs	age	female	marr	south	manufacturing	construction
0	5.10	0	8	21	35	1	1	0	1	0
1	4.95	0	9	42	57	1	1	0	1	0
2	6.67	0	12	1	19	0	0	0	1	0
3	4.00	0	12	4	22	0	0	0	0	0
4	7.50	0	12	17	35	0	1	0	0	0
5	13.07	1	13	9	28	0	0	0	0	0
6	4.45	0	10	27	43	0	0	1	0	0
7	19.47	0	12	9	27	0	0	0	0	0
8	13.28	0	16	11	33	0	1	0	1	0
9	8.75	0	12	9	27	0	0	0	0	0

```
In [5]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 534 entries, 0 to 533
Data columns (total 10 columns):
wage_per_hour     534 non-null float64
union             534 non-null int64
education_yrs     534 non-null int64
experience_yrs    534 non-null int64
age               534 non-null int64
female            534 non-null int64
marr              534 non-null int64
south             534 non-null int64
manufacturing     534 non-null int64
construction      534 non-null int64
dtypes: float64(1), int64(9)
memory usage: 41.8 KB
```

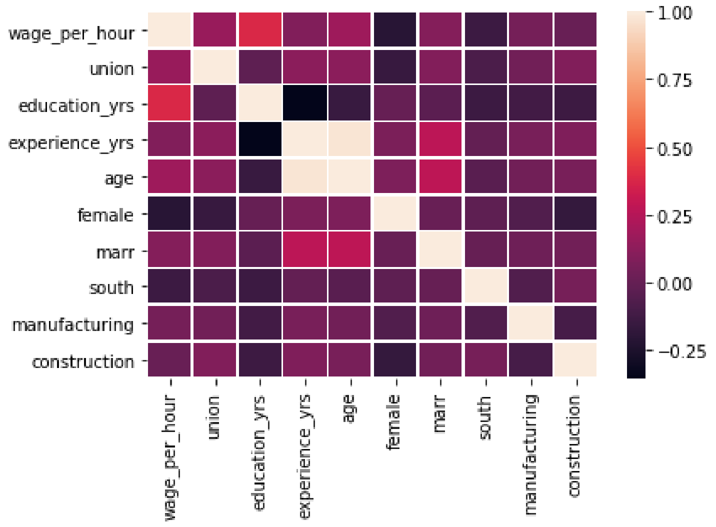
Model building steps

B. Understand the data (try to follow occam's razor: keep it simple)

```
In [26]: g = sns.pairplot(dataset[['wage_per_hour','education_yrs','experience_yrs','age','manufacturing','construction']])
```



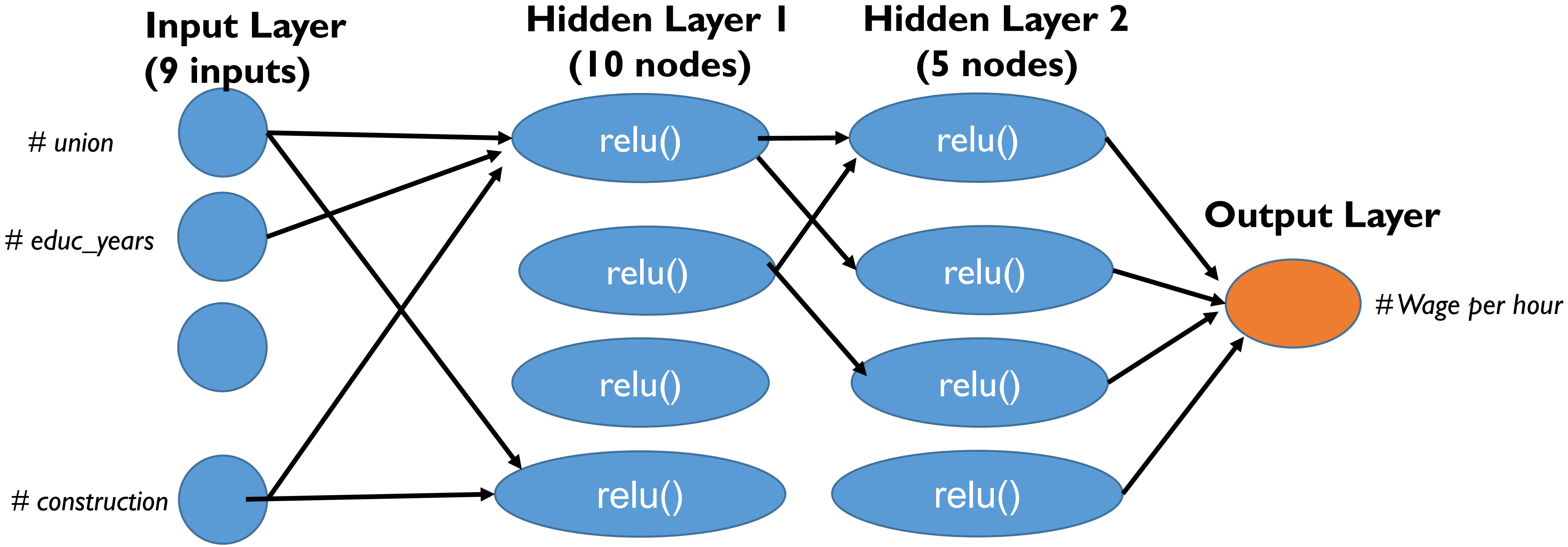

```
ax = sns.heatmap(correlation_matrix,linewidths=.5)
```



Model building steps

C. Specify the architecture of the model

```
In [33]: # Specify the model
         model=Sequential()
         model.add(Dense(10, activation='relu', input_shape = (numberofcolumns,)))
         model.add(Dense(5, activation='relu'))
         model.add(Dense(1))
```



Model building steps

D. Compile the model

```
In [36]: # Compile the model
         model.compile(optimizer='adam', loss='mean_squared_error',metrics=['mse','mae'])
```

In regression this is the most common loss function, we want our model to minimize the mean squared error

We want to observe the evolution of two metrics (mean squared error and mean average error) during the optimization process

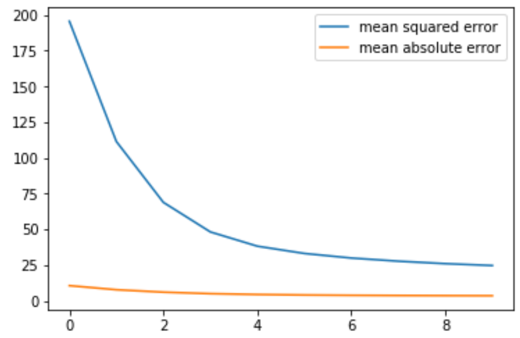
Model building steps

E. Fit the model

In [17]:
```
# fit model
history = model.fit(predictors,target,verbose=2,epochs=10)
```

```
Epoch 1/10
 - 0s - loss: 195.5772 - mean_squared_error: 195.5772 - mean_absolute_error: 10.6348
Epoch 2/10
 - 0s - loss: 111.4515 - mean_squared_error: 111.4515 - mean_absolute_error: 7.7783
Epoch 3/10
 - 0s - loss: 68.9013 - mean_squared_error: 68.9013 - mean_absolute_error: 6.0712
Epoch 4/10
 - 0s - loss: 48.2129 - mean_squared_error: 48.2129 - mean_absolute_error: 5.0308
Epoch 5/10
 - 0s - loss: 38.2373 - mean_squared_error: 38.2373 - mean_absolute_error: 4.4439
Epoch 6/10
 - 0s - loss: 33.1533 - mean_squared_error: 33.1533 - mean_absolute_error: 4.1025
Epoch 7/10
 - 0s - loss: 29.9454 - mean_squared_error: 29.9454 - mean_absolute_error: 3.8817
Epoch 8/10
 - 0s - loss: 27.7375 - mean_squared_error: 27.7375 - mean_absolute_error: 3.7444
Epoch 9/10
 - 0s - loss: 26.0194 - mean_squared_error: 26.0194 - mean_absolute_error: 3.6496
Epoch 10/10
 - 0s - loss: 24.7190 - mean_squared_error: 24.7190 - mean_absolute_error: 3.5691
```

In [18]:
```
# plot metrics
pyplot.plot(history.history['mean_squared_error'],label='mean squared error')
pyplot.plot(history.history['mean_absolute_error'],label='mean absolute error')
pyplot.legend()
pyplot.show()
```



Model building steps

F. Make predictions

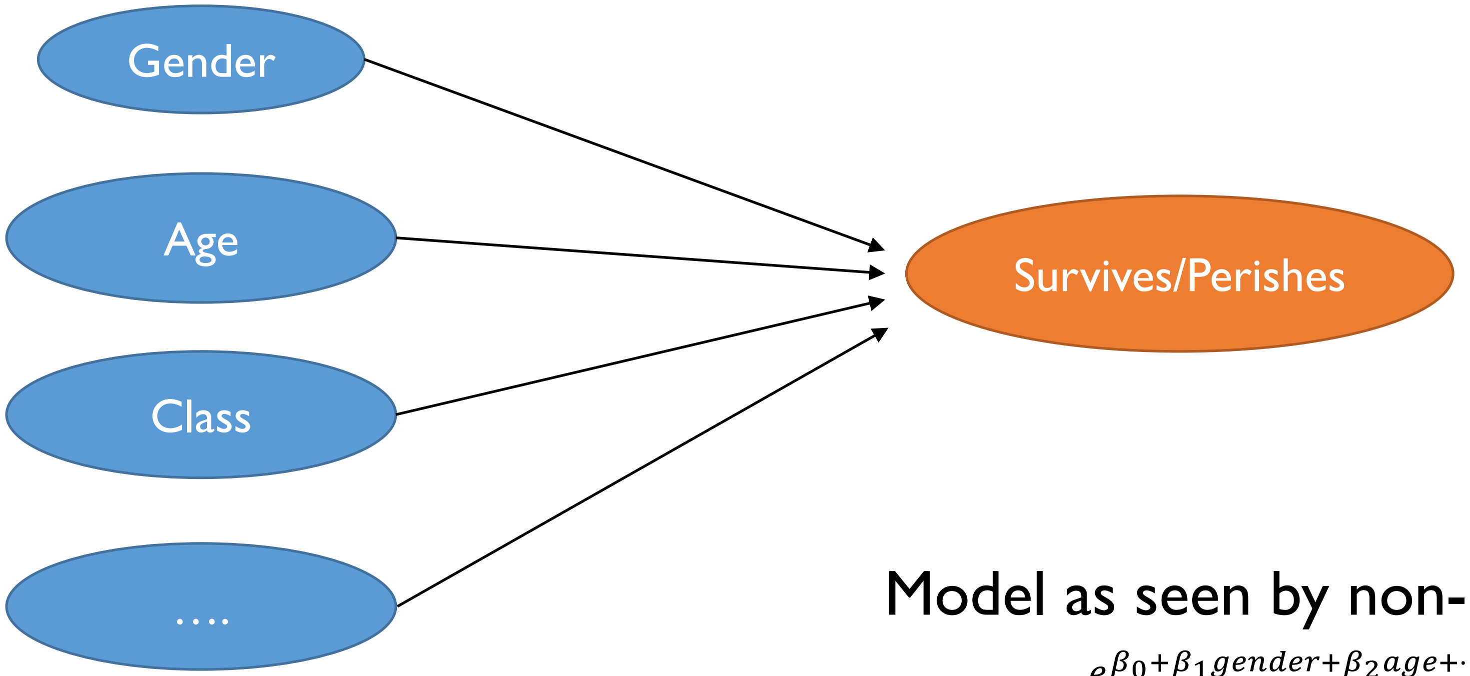
```
In [89]: model.predict(np.array([[0,8,21,35,1,1,0,1,0]]))

Out[89]: array([[ 7.06479549]], dtype=float32)

In [90]: model.predict(np.array([[0,12,4,22,0,0,0,0,0]]))

Out[90]: array([[ 7.32750416]], dtype=float32)
```

Let’s practice

IV. Classification example.

Imagine you work for AXA

- You need to predict if a titanic passenger will survive based on several features (gender, age, class, fare, cabin,..)

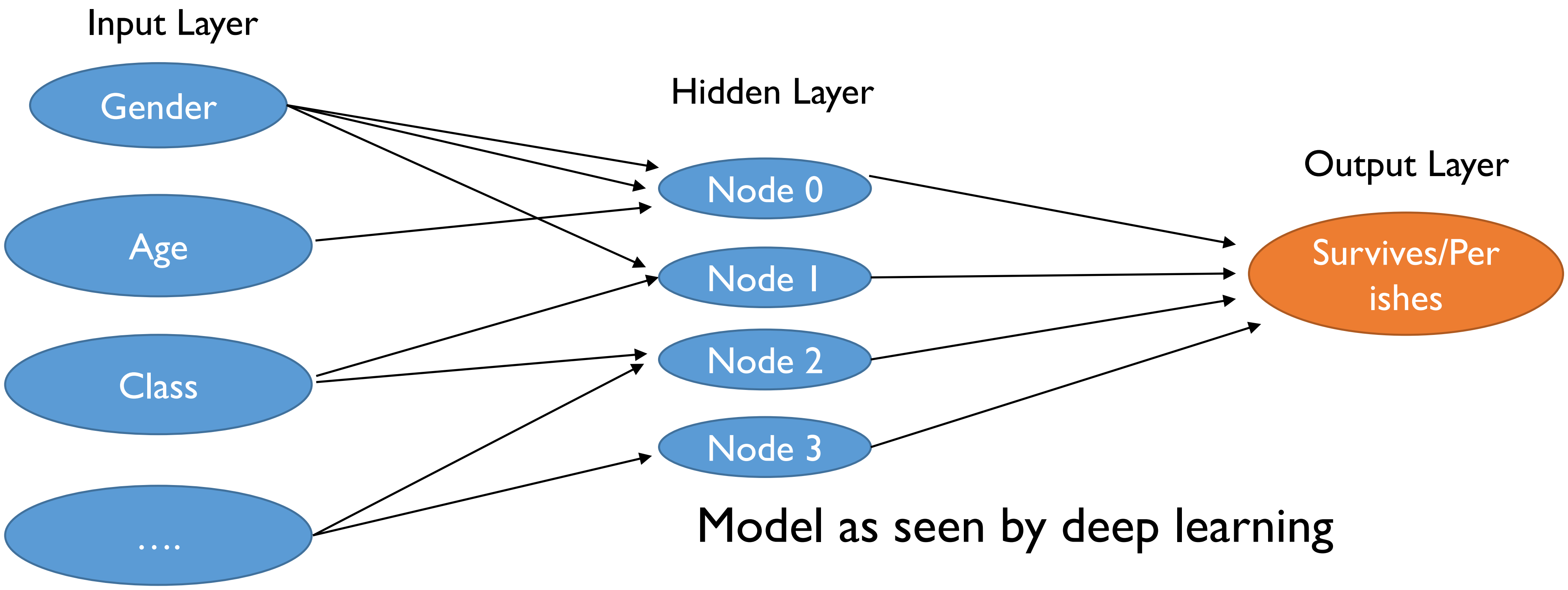


Model as seen by non-linear regression (logistic)

$$P(\text{Survival}) = \frac{e^{\beta_0+\beta_1 gender+\beta_2 age+\cdots}}{(1+e^{\beta_0+\beta_1 gender+\beta_2 age+\cdots})}$$

Imagine you work for AXA

- You need to predict how many transactions each customer will make next year.



Model as seen by deep learning

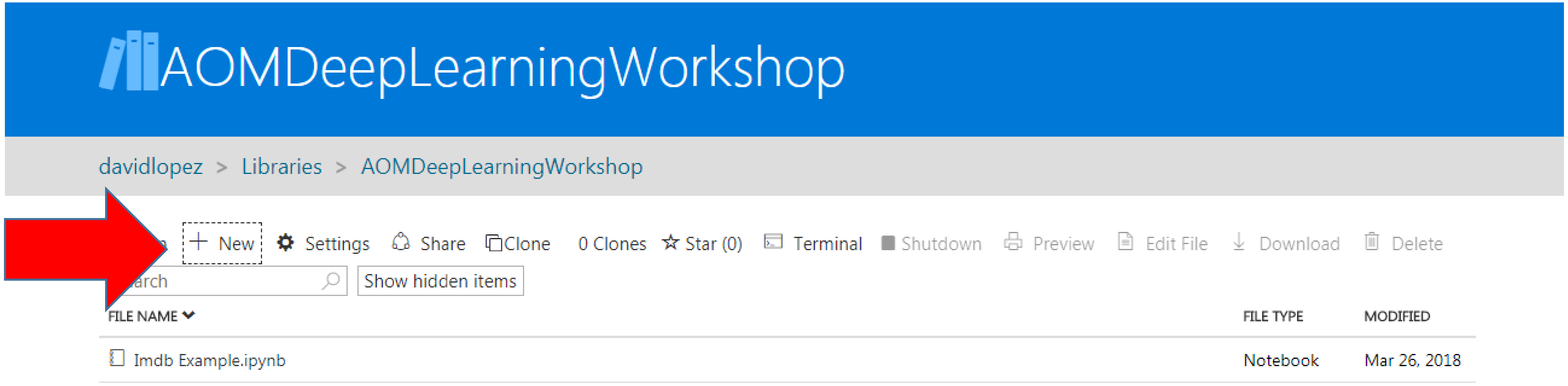
Model building steps

A. Import the data
B. Understand the data (exploratory analysis, clustering, correlation)
C. Specify Architecture of the model
D. Compile the model
E. Fit the model
F. Evaluate the model
G. Improve the model (more an art than a science as of today)
H. Make predictions

Imagine you work for AXA

- Let's import the code which we will be using in the next exercise.
- Go to your Azure library (e.g. AOMDeepLearningWorkshop)
- Import a Notebook. Click on "+ New", then "From URL", insert the following URL:

https://github.com/thousandoaks/AOMDeepLearningWorkshop/blob/master/Titanic.ipynb



Model building steps

A. Import the data

In [3]:
```
train_dataset=pd.DataFrame()
train_dataset=pd.read_csv('https://raw.githubusercontent.com/thousandoaks/AOMDeepLearningWorkshop/master/titanictrain.csv')
```

In [4]:
```
train_dataset.head(10)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

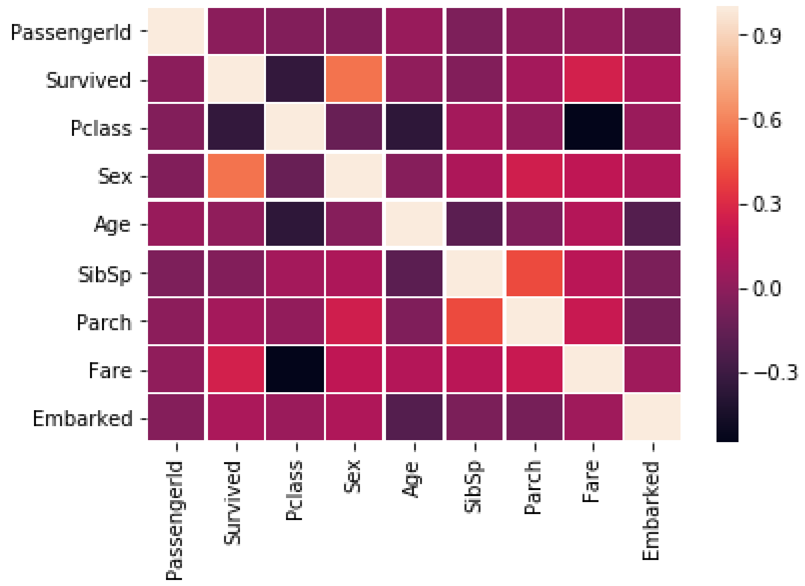
Model building steps

B. Understand the data

```
correlation_matrix=train_dataset.corr()
correlation_matrix
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
PassengerId	1.000000	-0.005007	-0.035144	-0.042939	0.038125	-0.057527	-0.001652	0.012658	-0.030467
Survived	-0.005007	1.000000	-0.338481	0.543351	0.010539	-0.035322	0.081629	0.257307	0.106811
Pclass	-0.035144	-0.338481	1.000000	-0.131900	-0.361353	0.083081	0.018443	-0.549500	0.045702
Sex	-0.042939	0.543351	-0.131900	1.000000	-0.024978	0.114631	0.245489	0.182333	0.116569
Age	0.038125	0.010539	-0.361353	-0.024978	1.000000	-0.184664	-0.048786	0.135516	-0.209388
SibSp	-0.057527	-0.035322	0.083081	0.114631	-0.184664	1.000000	0.414838	0.159651	-0.059961
Parch	-0.001652	0.081629	0.018443	0.245489	-0.048786	0.414838	1.000000	0.216225	-0.078665
Fare	0.012658	0.257307	-0.549500	0.182333	0.135516	0.159651	0.216225	1.000000	0.062142
Embarked	-0.030467	0.106811	0.045702	0.116569	-0.209388	-0.059961	-0.078665	0.062142	1.000000

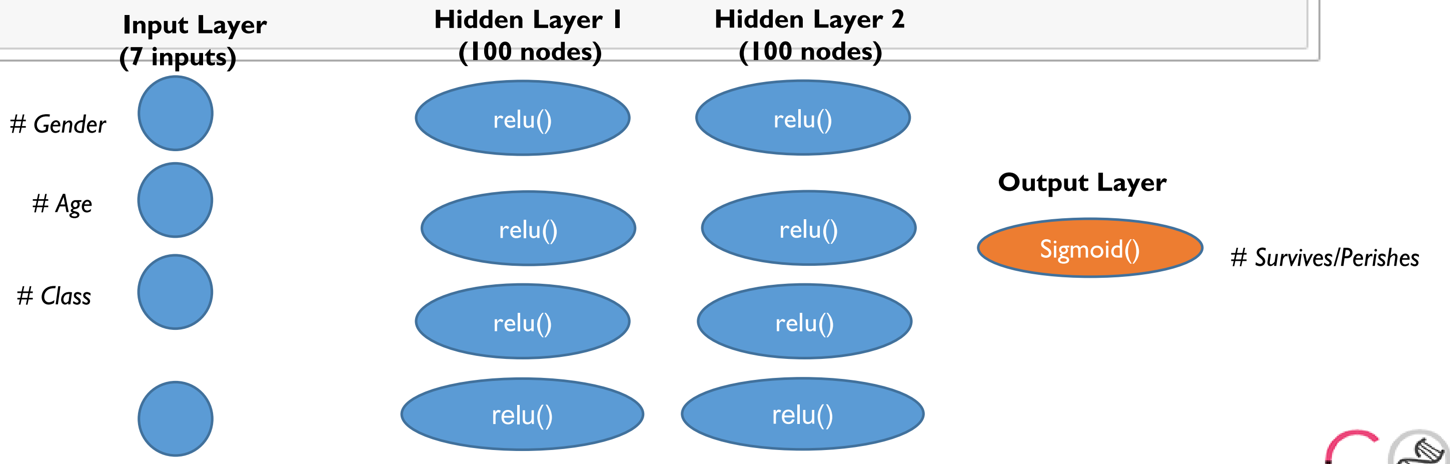
```
ax = sns.heatmap(correlation_matrix,linewidths=.5)
```



Model building steps

C. Specify the architecture of the model

```
In [15]: # Set up the model
         model = Sequential()

         # Add the first layer
         model.add(Dense(100,activation='relu',input_shape=(numberofcolumns,)))

         # Add a second layer
         model.add(Dense(100, activation='relu'))

         # Add the output layer
         model.add(Dense(1,activation='sigmoid'))
```



Model building steps

D. Compile the model

In [16]:
```
# Compile the model. Add accuracy as the metric to benchmark our model
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In binary classification this is the most common loss function, we want our model to minimize the classification error

We want to observe the evolution of accuracy, during the optimization process

Model building steps

We keep 10% of the data for validation purposes (overfitting control)

E. Fit the model

```
# Fit the model. This time we keep 20% of our samples for test purposes
model_training=model.fit(features,target,verbose=2,epochs=100,validation_split=0.1)
```

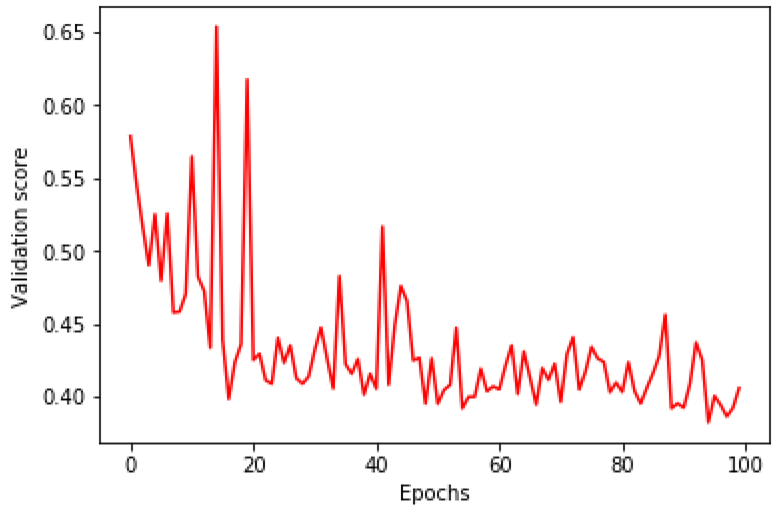
```
Train on 801 samples, validate on 90 samples
Epoch 1/100
 - 0s - loss: 0.8173 - acc: 0.6517 - val_loss: 0.5548 - val_acc: 0.7222
Epoch 2/100
 - 0s - loss: 0.6167 - acc: 0.6804 - val_loss: 0.5208 - val_acc: 0.8111
Epoch 3/100
 - 0s - loss: 0.5995 - acc: 0.6941 - val_loss: 0.5200 - val_acc: 0.7889
Epoch 4/100
 - 0s - loss: 0.6357 - acc: 0.6941 - val_loss: 0.6300 - val_acc: 0.7111
Epoch 5/100
 - 0s - loss: 0.7705 - acc: 0.6629 - val_loss: 0.5098 - val_acc: 0.7444
Epoch 6/100
 - 0s - loss: 0.7736 - acc: 0.6404 - val_loss: 0.6014 - val_acc: 0.7000
```

Train **Validation**

Model building steps

F. Evaluate the model

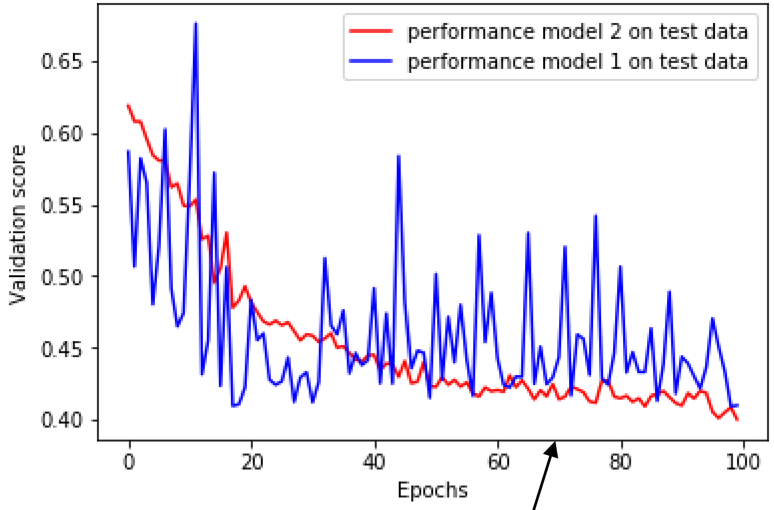
```
# Create the plot
pyplot.plot(model_training.history['val_loss'], 'r')
pyplot.xlabel('Epochs')
pyplot.ylabel('Validation score')
pyplot.show()
```



Model building steps

G. Improve the model

```
model2 = Sequential()
#input layer
model2.add(Dense(100, input_shape=(numberofcolumns,)))
model2.add(BatchNormalization())
model2.add(Activation("relu"))
model2.add(Dropout(0.4))

# hidden layers
model2.add(Dense(50))
model2.add(BatchNormalization())
model2.add(Activation("sigmoid"))
model2.add(Dropout(0.4))

model2.add(Dense(10))
model2.add(BatchNormalization())
model2.add(Activation("sigmoid"))
model2.add(Dropout(0.4))

#model2.add(Dense(2, activation="sigmoid"))

# output layer
model2.add(Dense(1, activation='sigmoid'))
```



Which model is better ?

Where to go next ?

Deep Learning resources

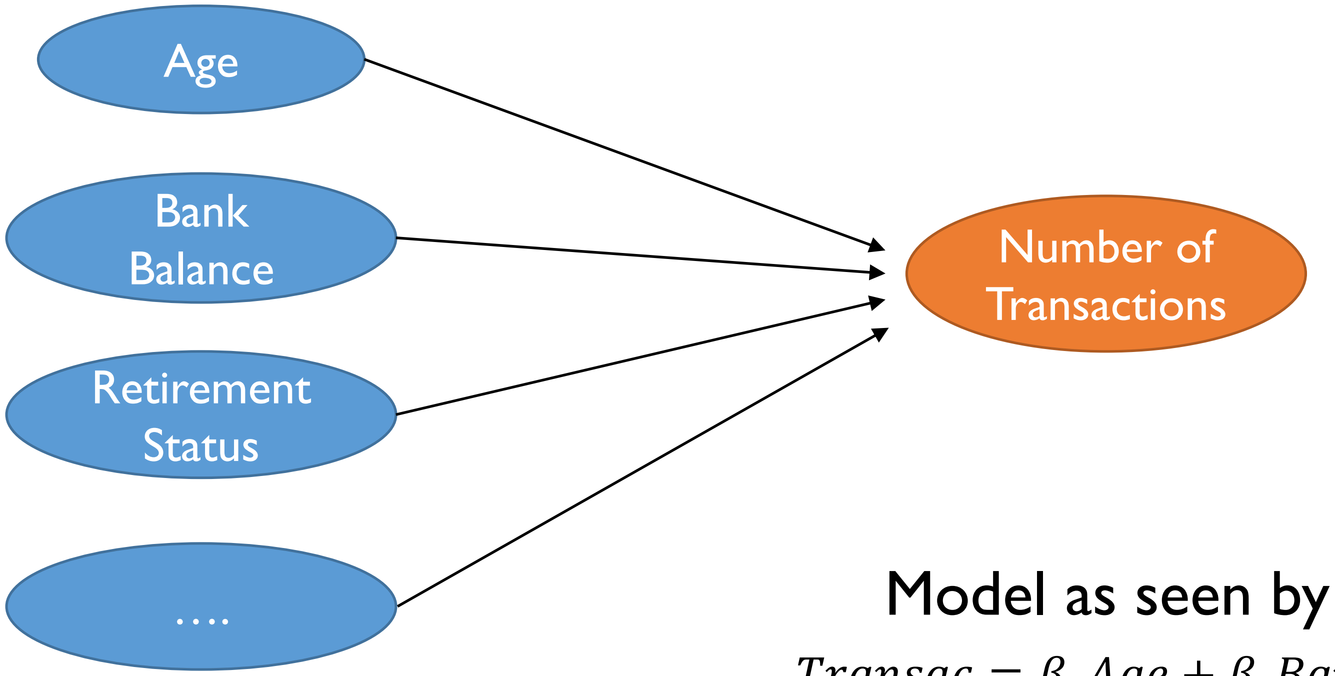
- https://keras.io/
- https://www.manning.com/books/deep-learning-with-python
- https://www.coursera.org/specializations/deep-learning (advanced)
- https://www.datacamp.com/courses/deep-learning-in-python (basic)
- https://www.kaggle.com/ (lots of projects and code to reuse)

Annex

Theory behind Deep Learning

Imagine you work for a bank

- You need to predict how many transactions each customer will make next year.

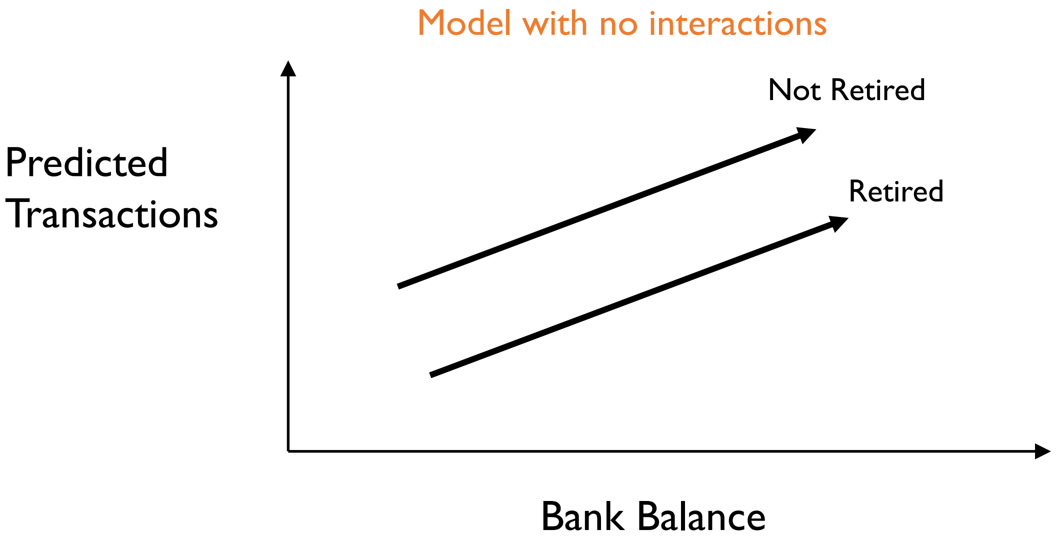


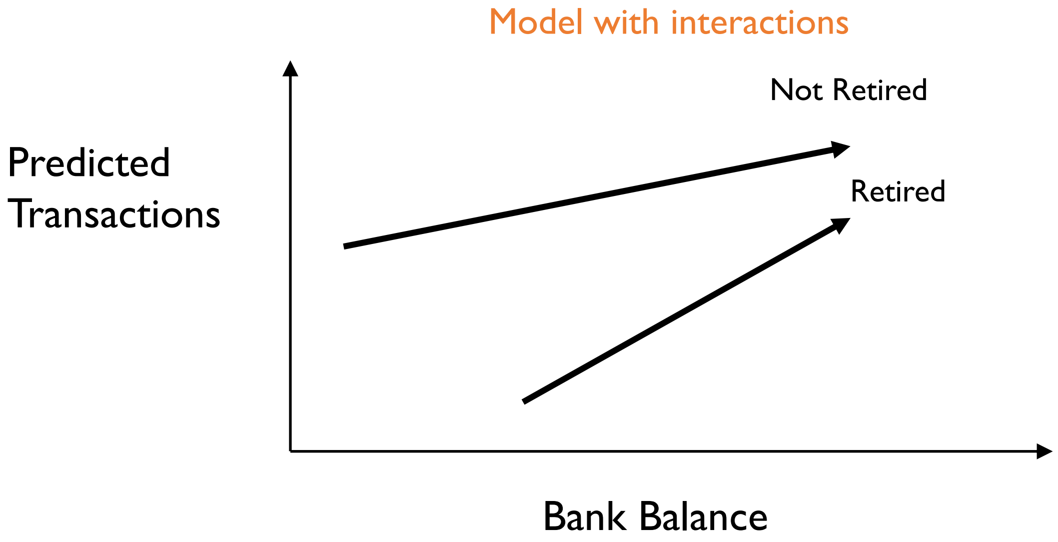
Model as seen by linear regression

$$Transac = \beta_0 Age + \beta_1 BankBalance + \beta_2\ Retired + \ldots$$

Imagine you work for a bank

- You need to predict how many transactions each customer will make next year.



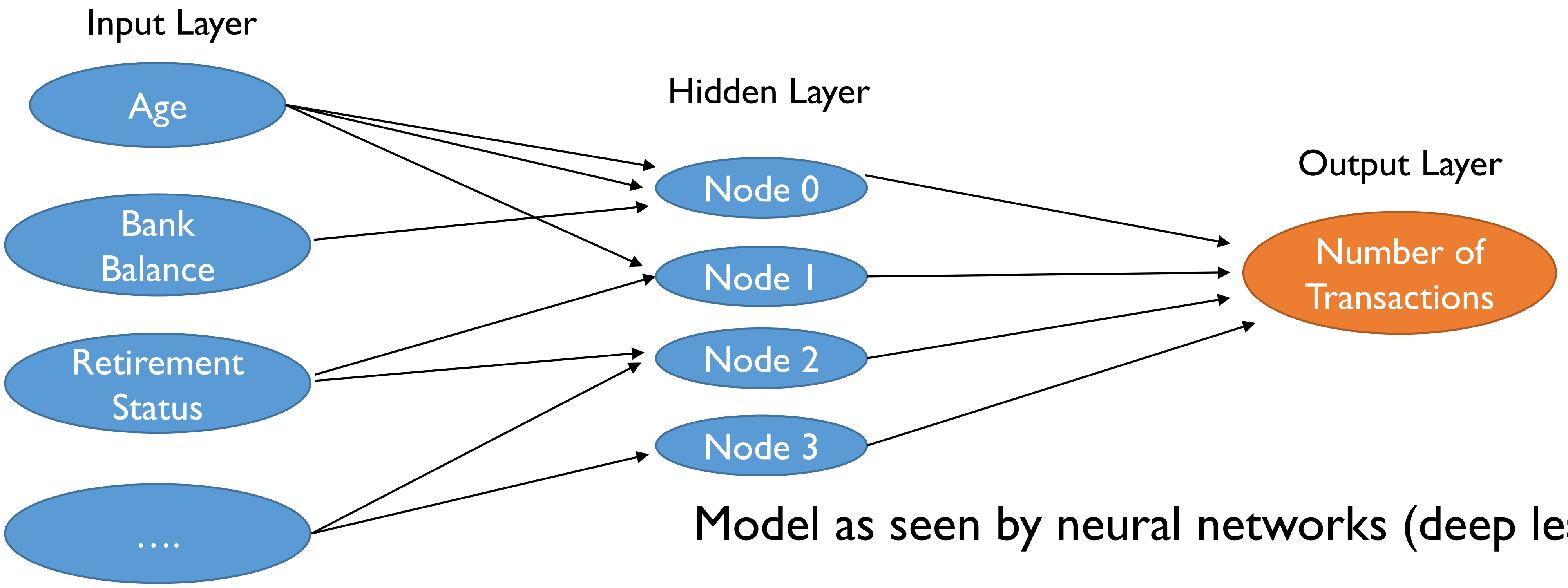


Model as seen by linear regression

$$Transac = \beta_0 Age + \beta_1 BankBalance + \beta_2\ Retired + \ldots$$

Imagine you work for a bank

- You need to predict how many transactions each customer will make next year.



Model as seen by neural networks (deep learning)

Interactions

- Neural networks account for interactions really well
- Deep learning uses especially powerful neural networks
  - Text
  - Images
  - Videos
  - Audio
  - Time series
  - Panel data

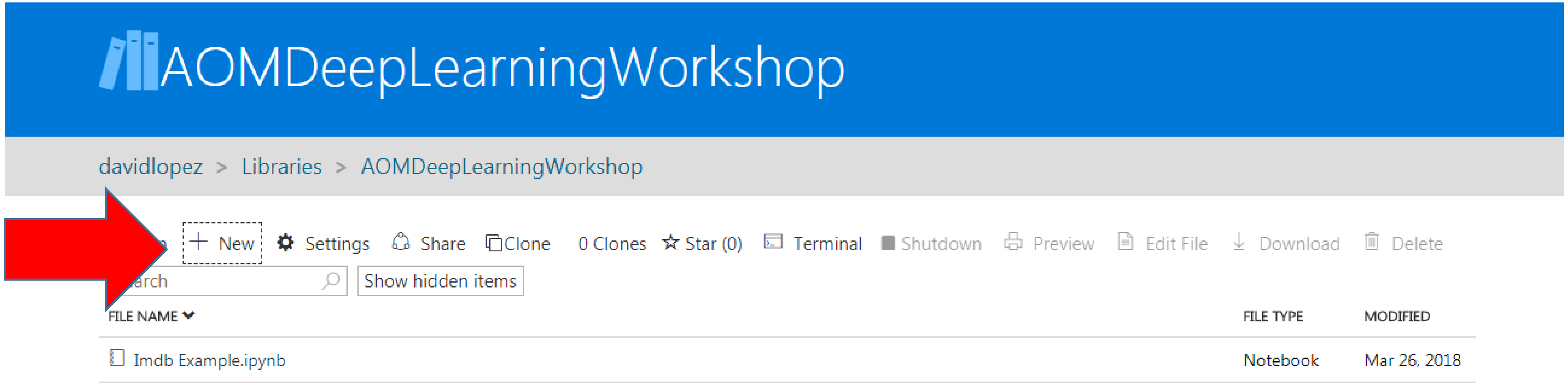
Annex

1. Forward propagation

Imagine you work for Netflix

- Let's import the code which we will be using in the next exercise.
- Go to your Azure library (e.g. AOMDeepLearningWorkshop)
- Import a Notebook. Click on "+ New", then "From URL", insert the following URL:

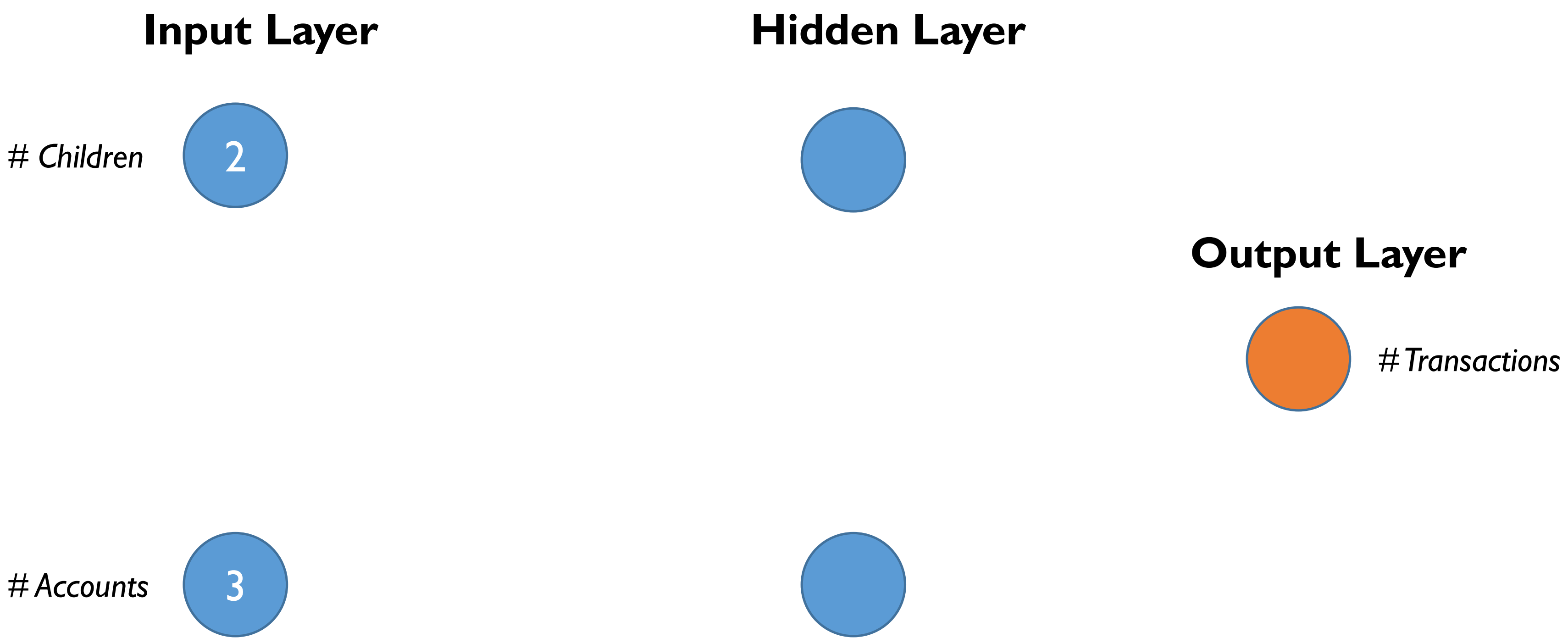
https://github.com/thousandoaks/AOMDeepLearningWorkshop/blob/master/Imdb%20Example.ipynb



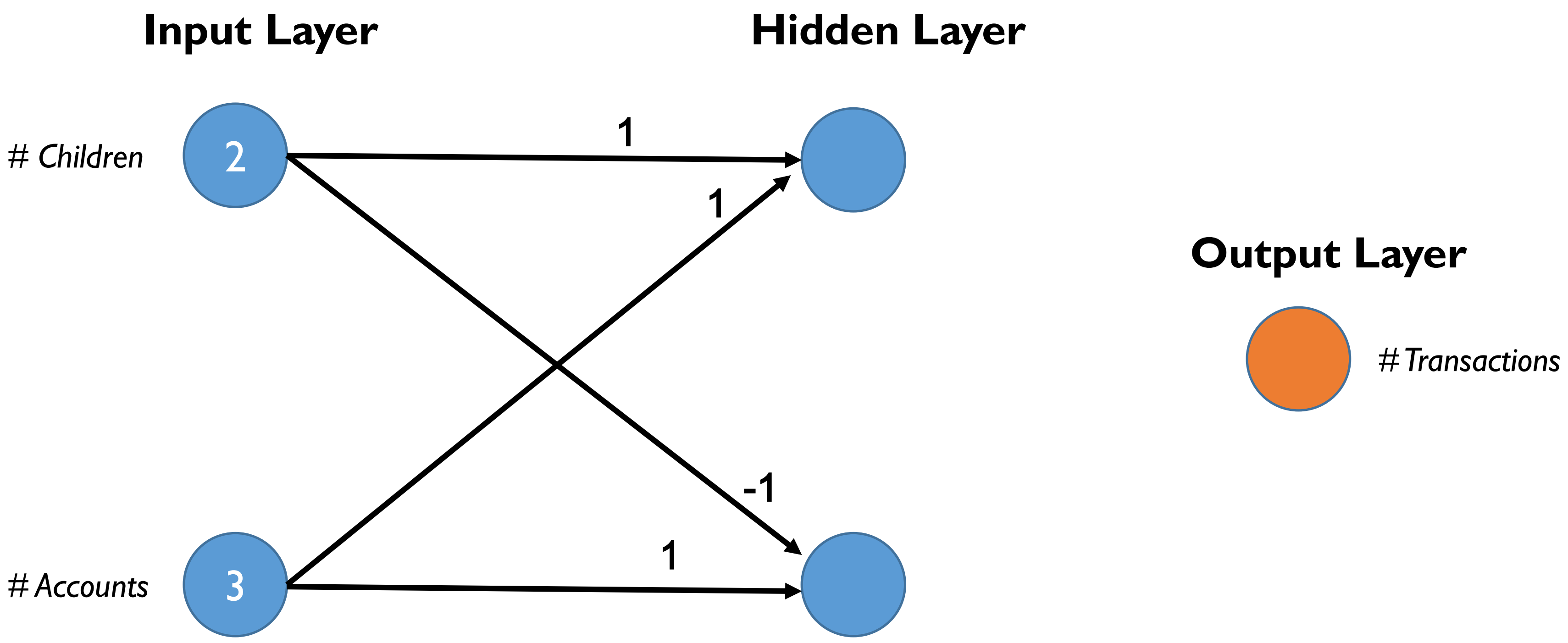
Bank transactions example

- Make predictions based on:
  - Number of children
  - Number of existing accounts

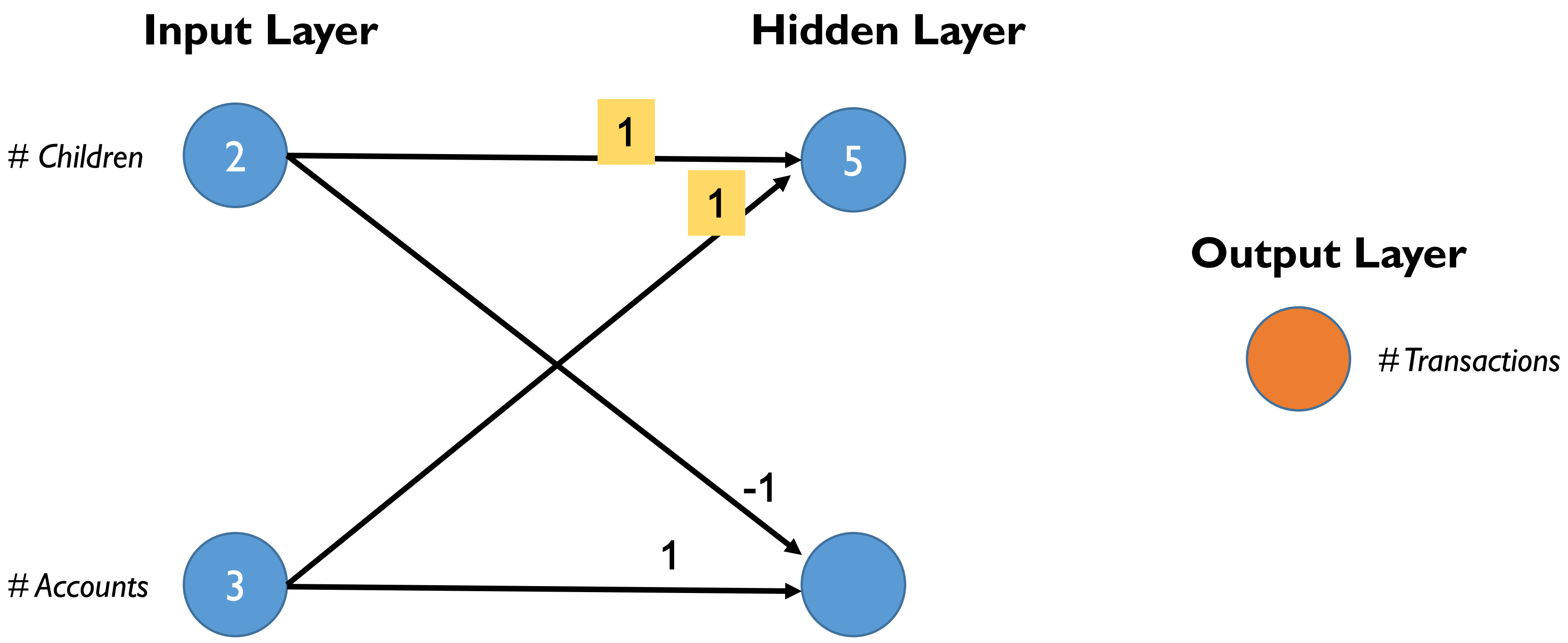
Forward propagation



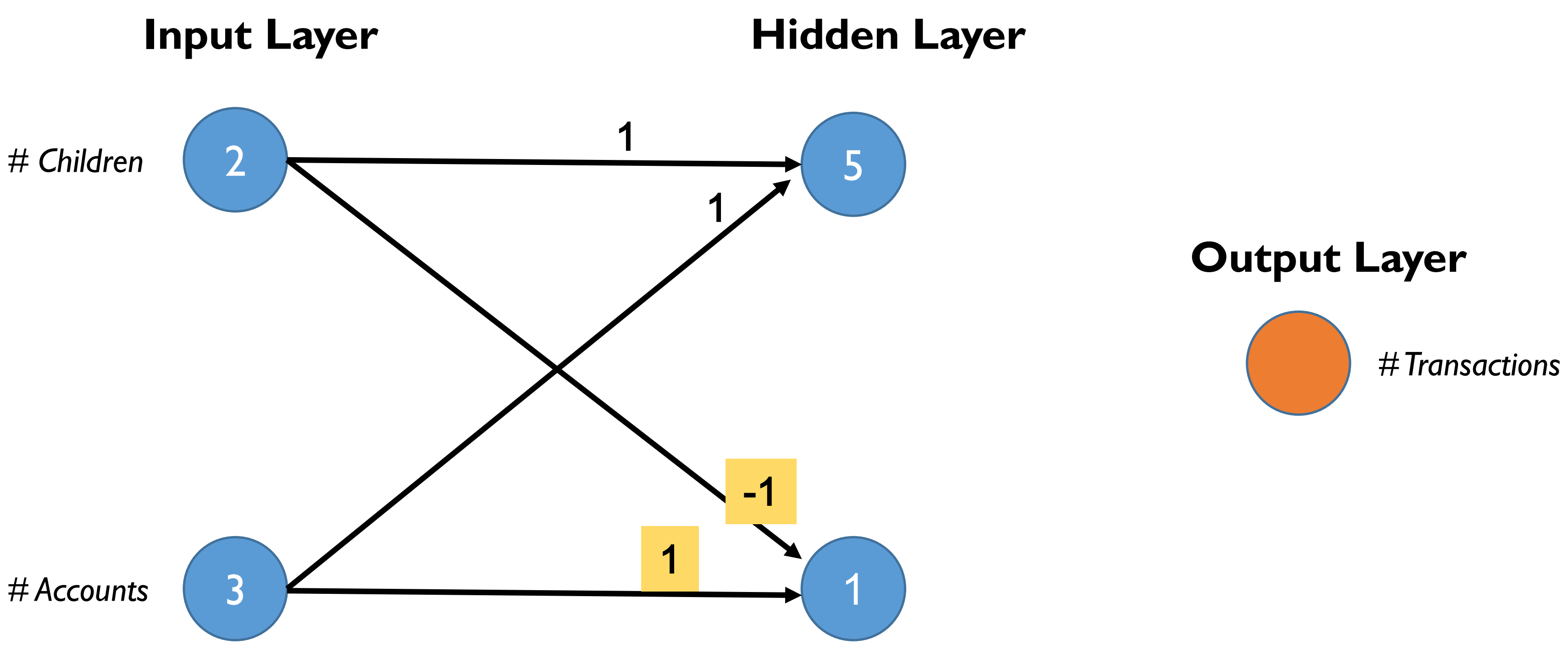
Forward propagation



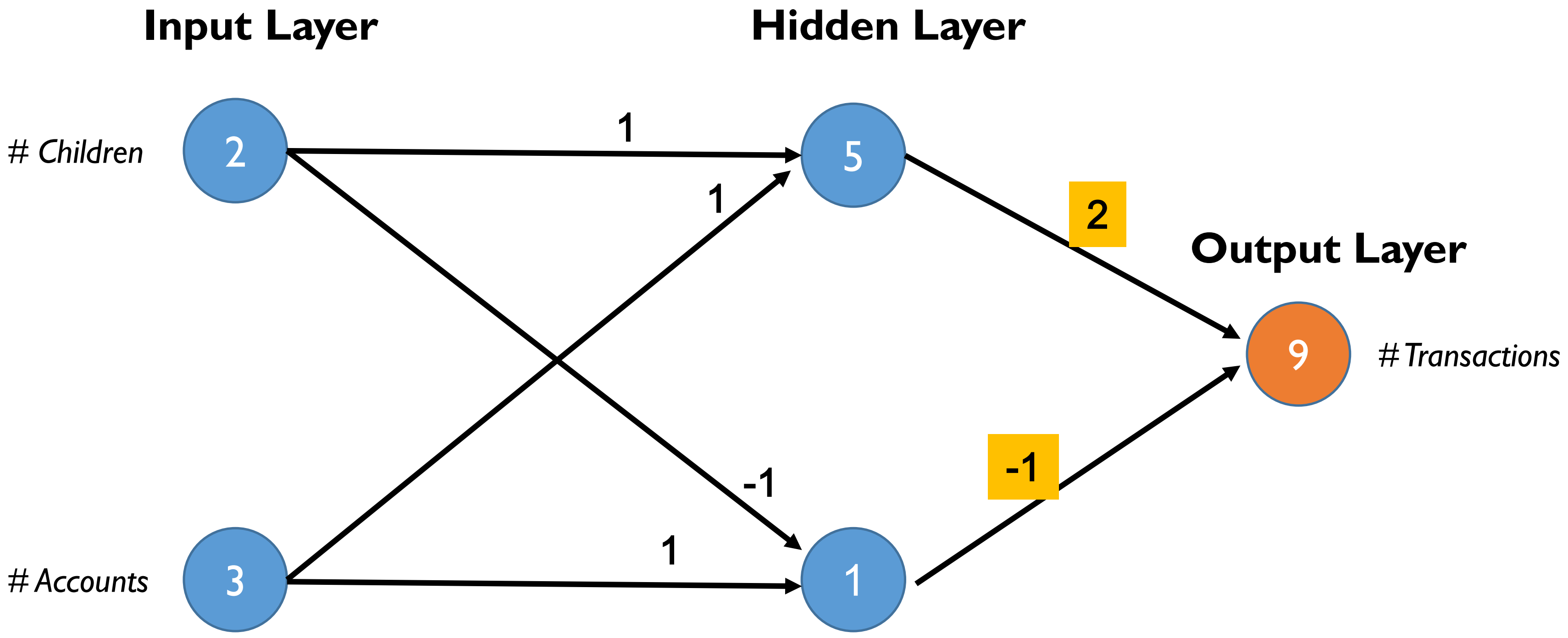
Forward propagation



Forward propagation



Forward propagation



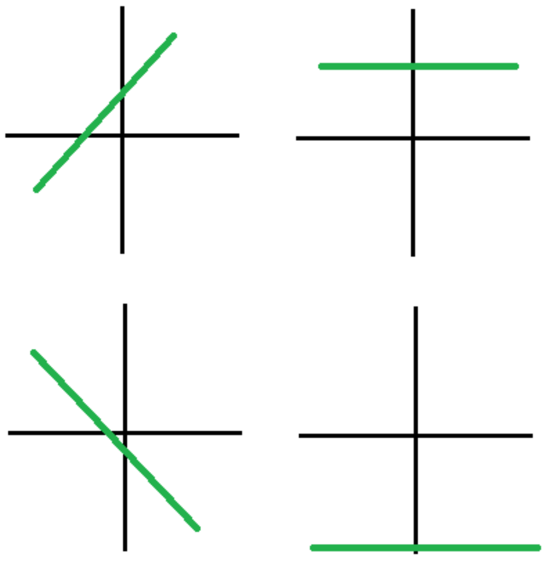
Forward propagation

- Multiply-add process
- Dot product
- Forward propagation for one data point at a time
- Output is the prediction for that data point

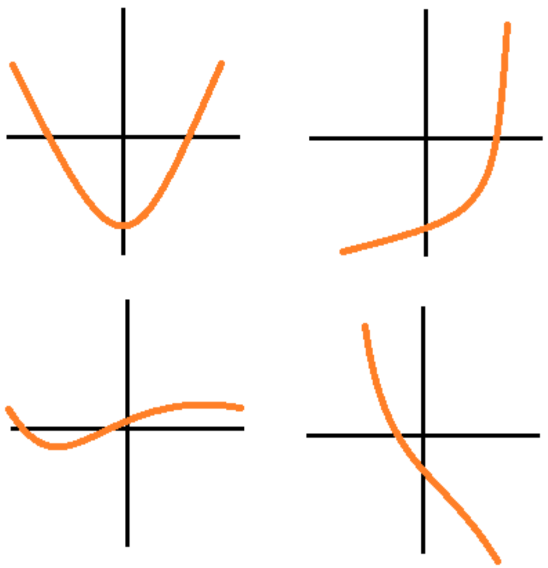
Annex

II. Activation functions

Linear vs Nonlinear Functions



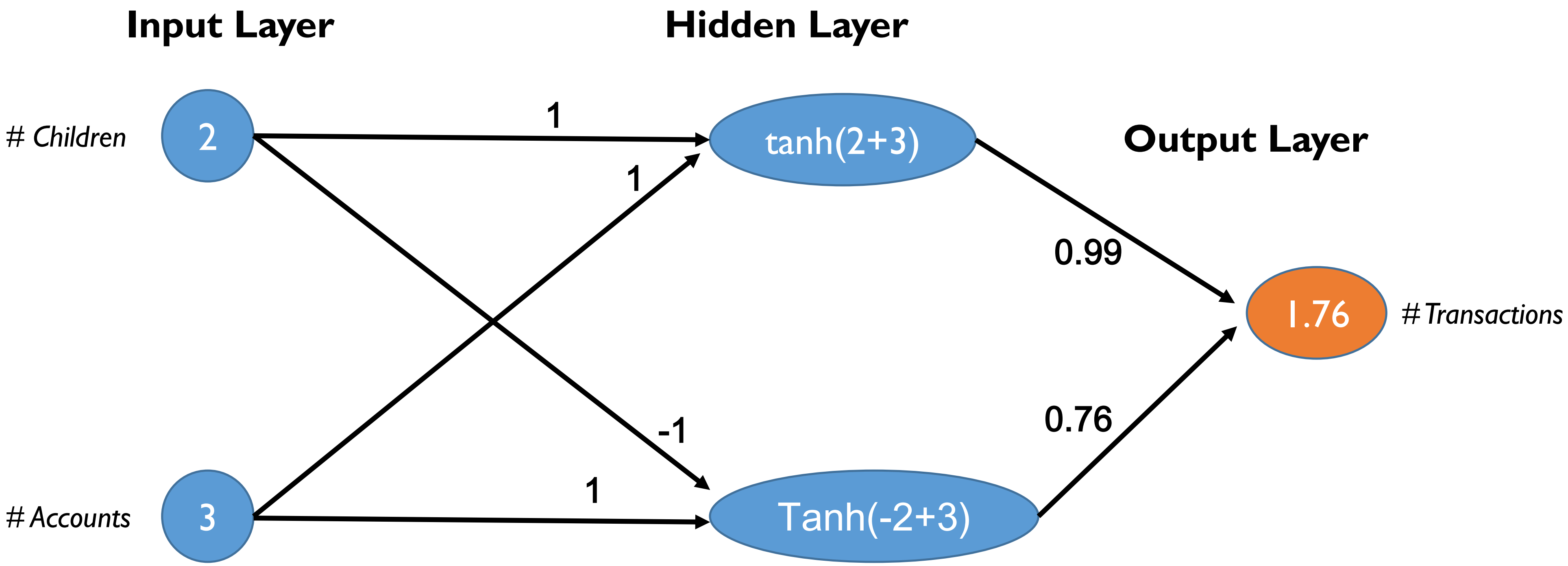
Linear Functions



Nonlinear Functions

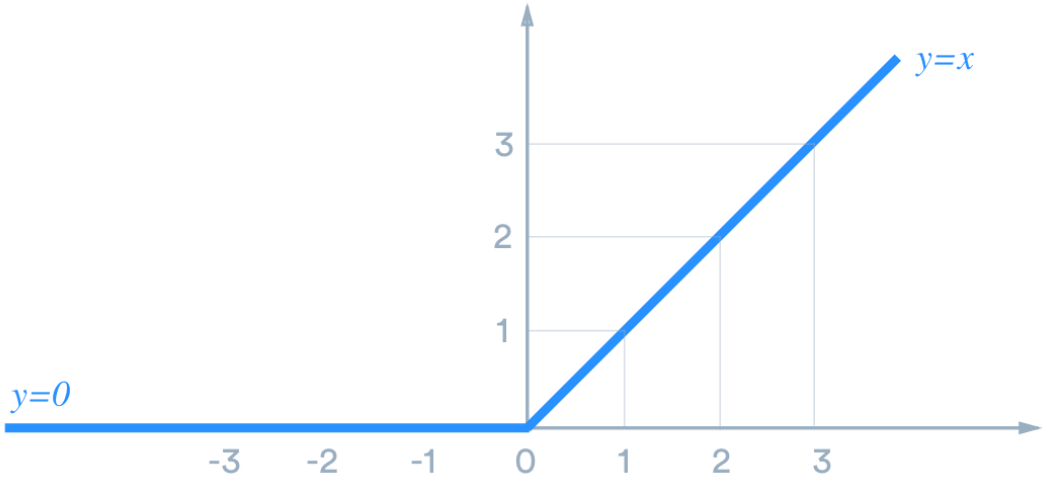
Activation Functions

- They allow us to include non-linearities in our model
- Applied to node inputs to produce node output



Activation Functions

- ReLU (Rectified Linear Activation) is the most common activation function



$$RELU(x) = \begin{cases} 0 \text{ } if \text{ } x < 0 \\ x \text{ } if \text{ } x >= 0 \end{cases}$$

Annex

III. Backward propagation

Backpropagation

- Applies optimization techniques (e.g. gradient descent) to update all weights in the network and minimize prediction error
- Computing and memory intensive (a lot) !!