# ON M'A RENTABILISÉ...
## deux Data Scientists témoignent !

Think Data. Think Ysance.

BIGDATA PARIS 2019 by Corp


Thomas Gorbinet
**Lead Data Scientist**

Aurélien Bénard
**Lead Data Scientist**


Lundi 11 mars 2019

Ysance

# Booster les ventes par l'IA...

## Comment mettre à disposition des équipes métiers B2C une solution efficace ?

# À propos de Ysance…

**Pour dynamiser vos conversions et vos ventes,**
nous vous apportons la maîtrise des parcours en simplifiant
la construction de votre patrimoine data réconciliée,
de la collecte à l'activation.

Ysance

**2005**
créé en

**250**
clients

**2**
implantations
à Lille & Paris

**15M€**
levée
de fonds

**160**
collaborateurs
dont 110 ingénieurs

PASS FRENCH TECH

**40**
personnes
dédiées à la R&D

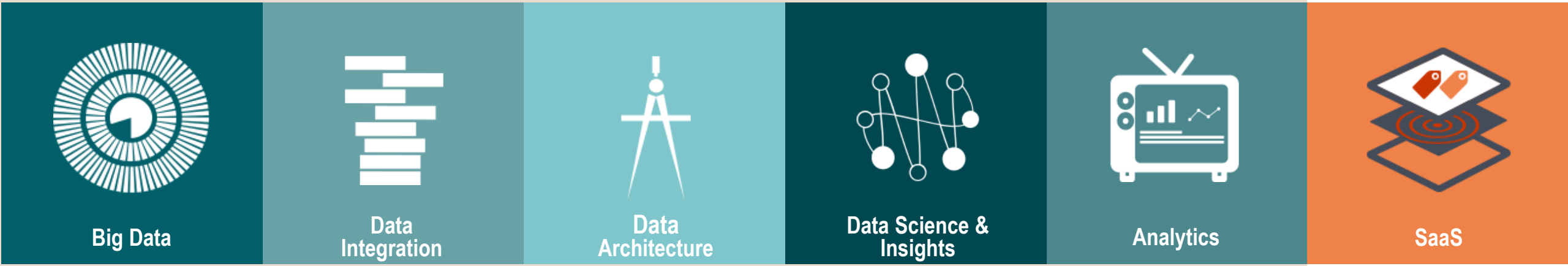
Gartner

FORRESTER

**711**
litres de
café / mois !

Think Data. Think Ysance.

# DEUX MÉTIERS, UN OBJECTIF




## Ysance exerce deux métiers complémentaires :

### l'intégration de plateformes data-centric & l'édition de solutions Saas pour le Marketing

- ***Data Services*** accompagne les entreprises dans leur transformation data-driven, couvrant l'ensemble de la chaîne de valeur de la data.

- ***Retail Marketing Platform*** apporte aux enseignes un marketing omnicanal agile qui augmente leur profitabilité et renforce la relation avec leurs clients.
- ***Attribution*** : optimisez vos coûts médias grâce à la compréhension de votre Customer Journey.
- ***Merchandising*** : donnez du sens à vos données avec la solution merchandising.

## Mon Directeur Marketing me demande de mettre à sa disposition une solution de prédiction des ventes…

- Quel coût pour mon projet d'IA ?
- Quels choix technologiques ?
- Comment lancer mon prototype en production ?
- Par où commencer ?
- Quelles sont les étapes clés pour réussir ma transformation IA ?
- De quelles compétences ai-je besoin?

Pierre P.
DSI

# Une vision holistique,
## du PoC à l'industrialisation.

**Bien démarrer.**

Lancer un premier cas d'usage et faire les bons choix technologiques au démarrage.

**Industrialiser sans peine.**

Passer du PoC à la mise en production à l'échelle.

**Quels bénéfices attendre ?**

Retours sur un cas d'usage concret.

# Commencez petit

## mais pensez sur toute la chaîne de valeur.

1 Choisir un cas d'usage réel et concret, qui répond à **des enjeux business précis.**

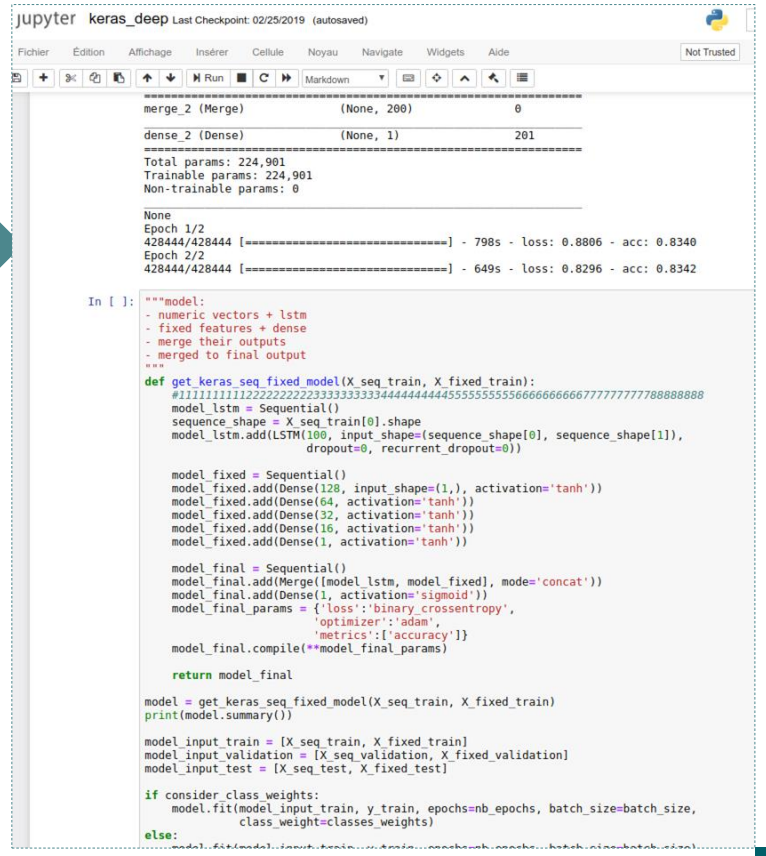
2 **Se restreindre à un PoC sur 60 jours/H maximum.**

3 **Définir les livrables attendus.**

4 **Impliquer l'ensemble des acteurs clés.**

5 **Envisager la mise en production des algorithmes.**

6 **Assurer un premier pas vers l'autonomie.**

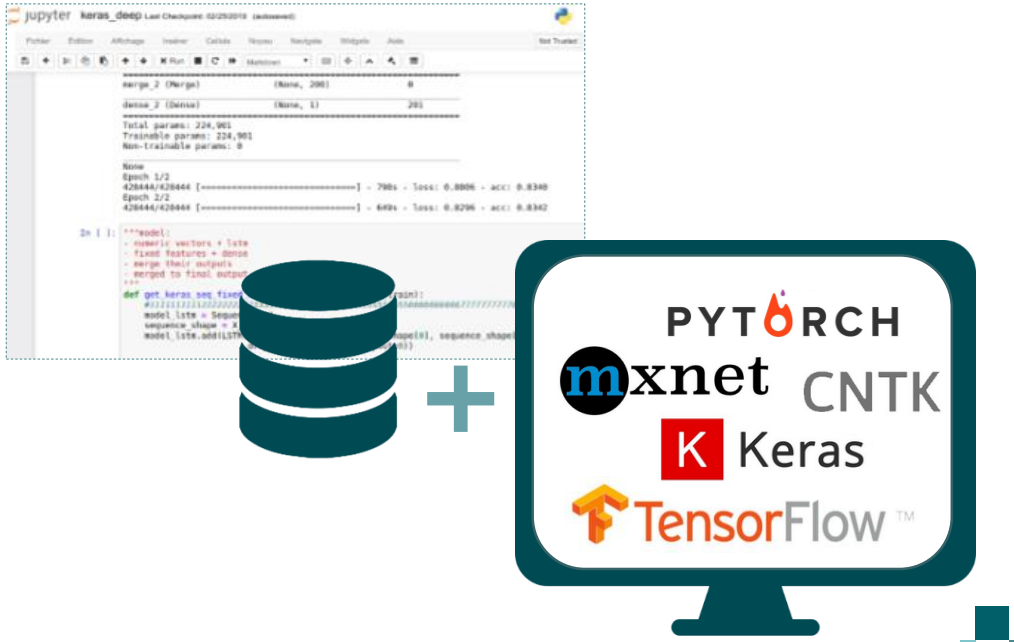



**Je veux booster mes ventes ?**
Et je code *on-premise*…

# Pensez grand

## même si vous commencez petit…


**Anticipez un déploiement sur le cloud**
lorsque vous développez *on-premise*…



# Cloud et IA
## sont indissociables.

**Capacité de stockage (+)**
**Puissance de calcul (+)**
**Gains de temps (+)**
**IA *on the shelf* ou sur-mesure (+)**
**Pas de maintenance (+)**
**Pas de gestion de parc (+)**
**informatique (+)**
**Souplesse (+)**
**Agilité (+)**

**Évaluation difficile des bénéfices/risques (-)**

# Les algorithmes IA
## pas facile non plus !

**Il existe beaucoup d'algorithmes…**
Quelques retours d'expériences.

**Algorithmes**

Utilisez de préférence un algorithme *on the shelf*.

**Arbitrages.**

Trouvez le compromis optimal entre interprétabilité & efficacité.

**Valeur.**

Focalisez-vous sur ce qui fait votre valeur sur le marché.

**ROI.**

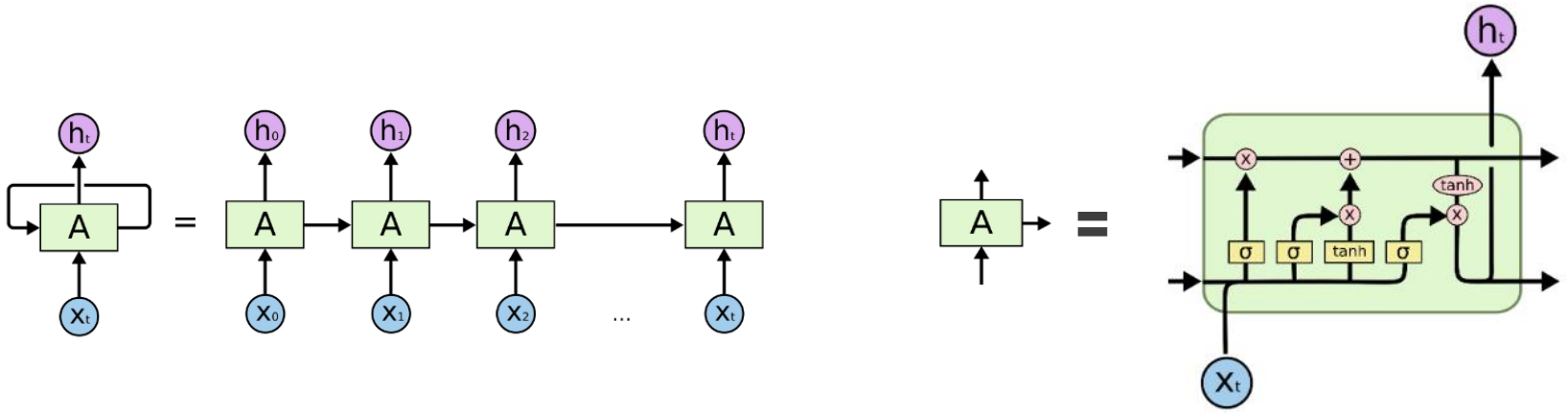
Trouvez le bon compromis entre budget & ROI.

**Ressources.**

Attirez des talents.

# L'algorithme
## de notre cas d'usage : la prédiction de la probabilité d'achat.

Modélisation de séquences d'événements.
**Architecture à mémoire** (LSTM).

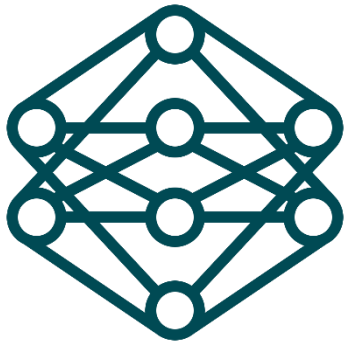


# De la théorie
## à la pratique.

Industrialisation :

- ☑ **Spécificité du déploiement cloud**
- ☑ **Orchestration**

**Réseau de neurones.**

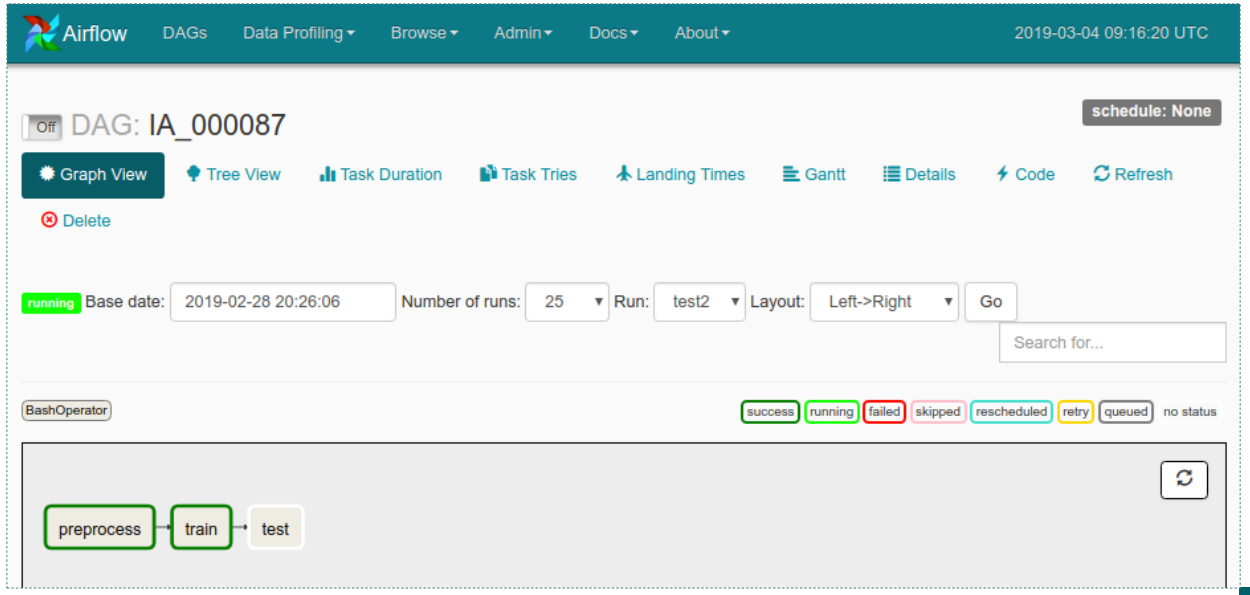


**Un module principal simple.**

*J'exécute une tâche T pour le compte C à la date D, en mode local (phase de développement, PoC) ou cloud (mode production.)*

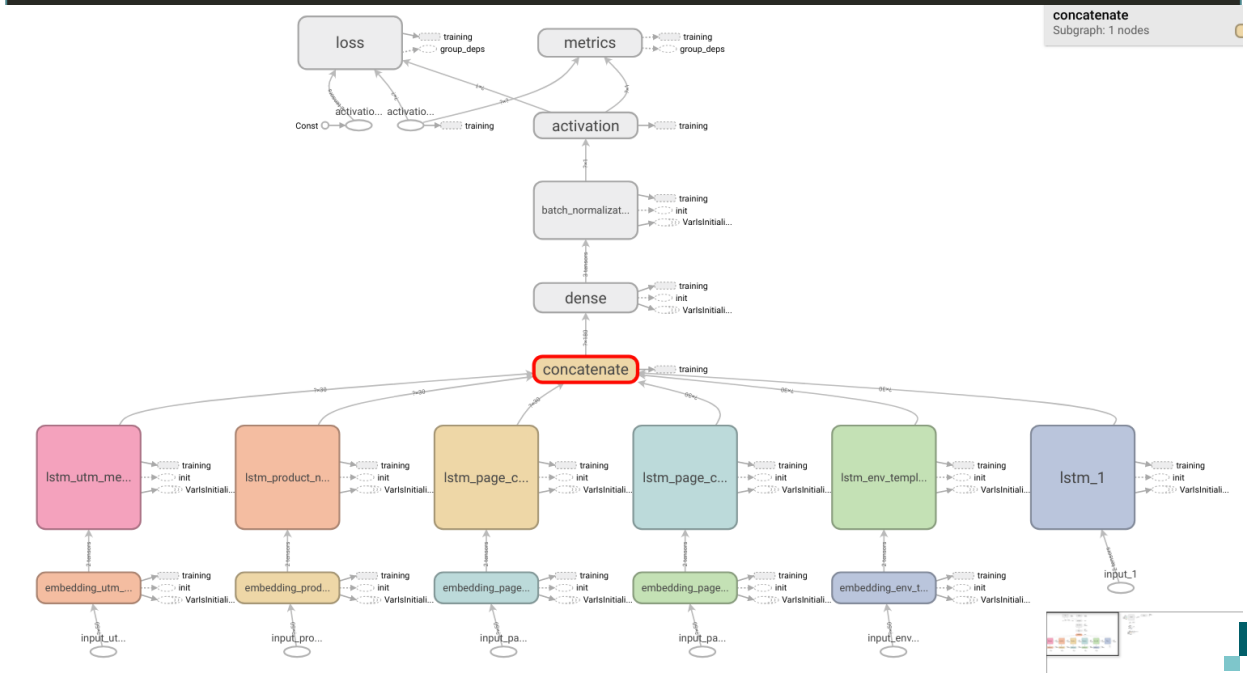
```
usage: main.py [-h] [--account_id ACCOUNT_ID] [--date DATE]
               [--task {preprocess,train,test,predict}] [--env {local,cloud}]
               [--conf CONF]

LSTM Network

optional arguments:
  -h, --help            show this help message and exit
  --account_id ACCOUNT_ID
                        account id to work on (default: None)
  --date DATE           airflow ds variable (default: None)
  --task {preprocess,train,test,predict}
                        task to perform (default: None)
  --env {local,cloud}   environment (default: local)
  --conf CONF           absolute or relative path to configuration file
                        (default: ../conf/dag.yaml)
```

**Un DAG (graphe orienté acyclique) basique.**
Trigger auto / extérieur.



# Une fonction pour définir le réseau de neurones. Keras

```python
def model(feat_seq, max_len, embed_shape, emb_dim, env):

    inputs = []
    models = []

    input_ = Input(shape=(max_len, len(feat_seq)), dtype='float32', name='input_1')
    lstm_ = CuDNNLSTM(30, name='cudnnlstm_1')(input_) if env=='cloud' else LSTM(30, name='lstm_1')(input_)

    inputs.append(input_)
    models.append(lstm_)

    for key, value in embed_shape.items():
        input_ = Input(shape=(max_len,), dtype='float32', name='input_%s'%key)
        inputs.append(input_)
        embedding_ = Embedding(value, emb_dim, input_length=max_len, name='embedding_%s'%key)(input_)
        lstm_ = CuDNNLSTM(30, name='cudnnlstm_%s'%key)(embedding_) if env=='cloud' else LSTM(30, name='lstm_%s'%key)(embedding_)
        models.append(lstm_)

    models_merged = Concatenate(axis=1)(models)

    output = Dense(1, use_bias=False)(models_merged)
    output = BatchNormalization()(output)
    output = Activation("sigmoid")(output)

    model_params = {'loss':'binary_crossentropy',
                    'optimizer':'adam',
                    'metrics':['acc']}
    model = Model(inputs=inputs, outputs=output)
    model.compile(**model_params)

    logger.info(model.summary())

    return model
```



# Une fonction pour le prétraitement des données. Pandas (python)

```python
def preprocess(self):

    logger.info("PREPROCESSING...")

    df = self.gpie.convey(source='gs', destination='dataframe', data_name=param.df)

    df.loc[:, 'hit_timestamp'] = df.hit_timestamp.apply(
        lambda x: datetime.strptime(str(x), '%Y-%m-%d %H:%M:%S UTC'))

    df = df[[param.key, param.event_timestamp, param.target[0]]+param.embedding+param.dummy+param.numerical]

    df_sorted = df.sort_values([param.key, param.event_timestamp], ascending=[True, True], inplace=False)
    df_sorted.reset_index(drop=True, inplace=True)

    # EMBEDDING
    df_label_encoded = pd.DataFrame()
    for feature in param.embedding:
        values = df_sorted[feature].unique()
        encoding_labels = np.arange(values.shape[0])+1
        dict_labels = dict(zip(values,encoding_labels))
        df_label_encoded[feature] = df_sorted[feature].apply(lambda x: dict_labels[x])
    df_sorted.drop(param.embedding, inplace=True, axis=1)

    # DUMMY
    df_dummy = pd.get_dummies(df_sorted[param.dummy])
    df_sorted.drop(param.dummy, inplace=True, axis=1)

    # DELTA TIME
    serieDeltaTime = df_sorted[param.event_timestamp].diff()
    mask = df_sorted.master_id != df_sorted.master_id.shift(1)
    serieDeltaTime[mask] = timedelta(0)
    serieDeltaTime  = serieDeltaTime.astype('timedelta64[s]')
    df_delta_time = serieDeltaTime.to_frame(name='delta_time')
    df_sorted.drop(param.event_timestamp, inplace=True, axis=1)

    # CONCAT
    dfs_concat = (df_label_encoded,
                  df_dummy,
                  df_delta_time,
                  df_sorted)
    df = pd.concat(dfs_concat, axis=1)
    df = df.fillna(-1)

    self.gpie.convey(source='dataframe', destination='gs', dataframe=df, data_name=param.df_preprocessed)

    logger.info("PREPROCESSING OK")
```

## Une fonction pour l'entraînement :

- séparation train / validation / test,
- instanciation du modèle,
- reshaping des variables d'entrée,
- ajout d'un callback pour TensorBoard.

```python
def train(self, verbose=2):
    pass

    logger.info("TRAINING...")

    df = self.gpie.convey(source='gs', destination='dataframe', data_name=param.df_preprocessed, delete_in_gs=False)

    df = df[df.master_id.str.contains(param.regex['train'])]
    df.reset_index(drop=True, inplace=True)

    embed_shape = {}
    for feature in param.embedding:
        embed_shape[feature] = int(np.max(df[feature].values))+1

    master_ids_df = df.master_id.unique()
    _ = random.shuffle(master_ids_df)

    y = rnn.get_target(df, param.target)
    df.drop(param.target[0], inplace=True, axis=1)

    feat_seq = list(set(df.columns) - set([param.key, param.event_timestamp] + param.embedding))

    X_seq, dict_Xs_emb = rnn.shape_data(df, set(master_ids_df), feat_seq, param.embedding, param.max_len)

    model = rnn.model(feat_seq, param.max_len, embed_shape, param.emb_dim, self.env)

    model_inputs = [X_seq] + [value for key, value in dict_Xs_emb.items()]
    classes_weights = {0 : 1., 1 : y.shape[0]/y.sum()-1}
    logger.info(classes_weights)

    tbCallBack = TrainValTensorBoard(log_dir='gs://%s/%s/graph'%(self.bucket_id, param.bucket_path))

    model.fit(model_inputs, y, epochs=param.epochs,
                               batch_size=param.batch_size,
                               class_weight=classes_weights,
                               validation_split=0.15,
                               # callbacks=[tbCallBack],
                               verbose=verbose)

    # SAVE MODEL
    model_json = model.to_json()
    with open("/tmp/%s.json"%param.model_name, "w") as json_file:
        json_file.write(model_json)
    model.save_weights("/tmp/%s.h5"%param.model_name)
    self.gpie.convey(source='local', destination='gs', data_name=param.model_name)

    predictions = model.predict(model_inputs, verbose=0)

    logger.info("BRIER SCORE:  %f"%brier_score_loss(y, predictions))

    fpr, tpr, thresholds = roc_curve(y, predictions)
    logger.info("AUC: %f"%auc(fpr, tpr))

    fig = display.get_roc_curve(y, predictions)
    fig_name = "%s_%s.png"%(param.model_name, 'train')
    fig.savefig("%s/%s"%(param.local_dir_path, fig_name), bbox_inches='tight')
    self.gpie.convey(source='local', destination='gs', data_name=fig_name)

    logger.info("TRAINING OK")
```

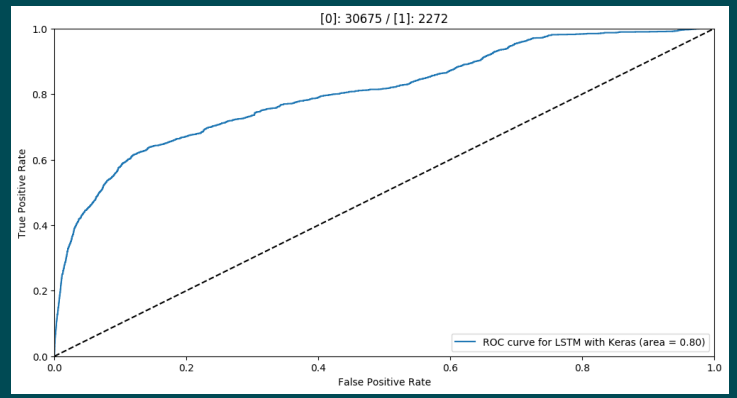
## Une fonction pour instancier un job ML Engine :

- pythonVersion,
- runtimeVersion (TensorFlow),
- masterType / workerType (large_model, standard_gpu…)

main.py

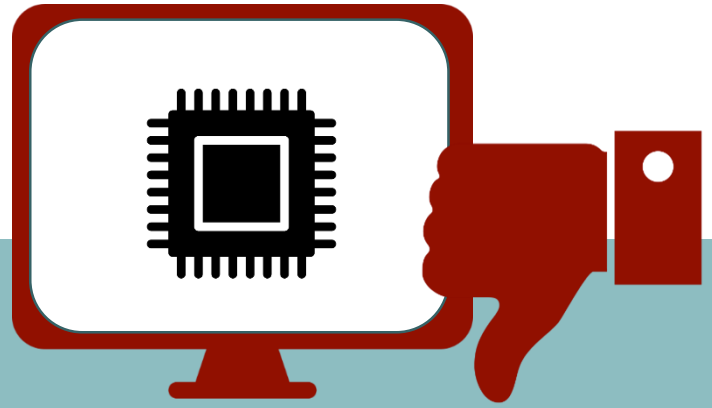
```
60
61  def ml_job(ml_engine_service, project_account, bucket_id, task):
62
63      job_parent = "projects/{project}".format(project=project_account)
64      now_str = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
65      job_id = "job_{}_{}".format(account_id, now_str)
66
67      job_body = {'trainingInput':
68                  {'pythonVersion': param.ml_pythonVersion,
69                   'runtimeVersion': param.ml_runtimeVersion,
70                   'scaleTier': param.ml_typology[task]['ml_scaleTier'],
71                   'region': param.ml_region,
72                   'pythonModule': 'model.model',
73                   'args': ["--project_account", project_account,
74                            "--bucket_id", bucket_id,
75                            "--task", task],
76                   'packageUris': [
77                     "gs://%s/%s/mymodel-0.0.1-py3-none-any.whl"%(bucket_id, param.bucket_path),
78                     "gs://%s/%s/google_pandas_import_export-1.1rc0-py3-none-any.whl"%(bucket_id, param.bu
79                   ],
80                   'masterType': param.ml_typology[task]['ml_masterType']
81                   },
82                  'jobId': job_id}
83
84      logging.info("job_body: %s" % job_body)
85      logging.info("job_parent: %s" % job_parent)
86      logging.info("creating a job ml: %s" % job_id)
87      return ml_engine_service.projects().jobs().create(parent=job_parent, body=job_body), job_id
88
89
90  if __name__ == '__main__':
91
92      logging.getLogger('urllib3').setLevel(logging.WARNING)
93      logging.getLogger('google').setLevel(logging.WARNING)
94      logging.getLogger('googleapiclient').setLevel(logging.WARNING)
95      logging.getLogger('google_auth_httplib2').setLevel(logging.WARNING)
96      logging.getLogger('googleapiclient.discovery_cache').setLevel(logging.ERROR)
97
98      parser = argparse.ArgumentParser(description="LSTM Network",
99                                       formatter_class=argparse.ArgumentDefaultsHelpFormatter)
100     parser.add_argument("--account_id", dest="account_id", help="account id to work on")
101     parser.add_argument("--date", dest="date", help="airflow ds variable")
102     parser.add_argument("--task", dest="task", choices=['preprocess', 'train', 'test', 'predict'], help='
```

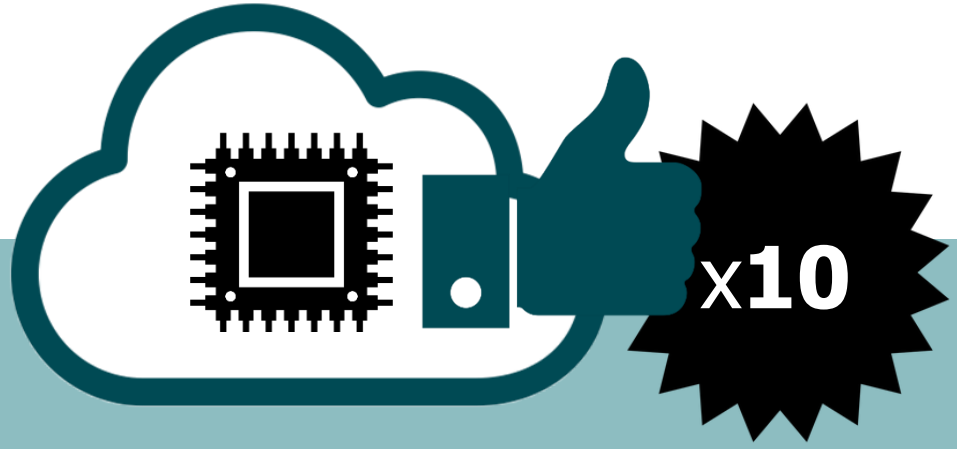
# Apprentissage et évaluation du réseau

# GPU rocks!

## CPU vs GPU






**Utilisation CPU**

**Utilisation GPU**

```
Epoch 1/64
2019-03-01 14:13:27.541559: I tensorflow/core/platform/cpu_feature_guard.cc:
hat this TensorFlow binary was not compiled to use: AVX2 FMA
   141s - loss: 0.4626 - acc: 0.6853 - val_loss: 0.5553 - val_acc: 0.8651
Epoch 2/64
 - 136s - loss: 0.3839 - acc: 0.7719 - val_loss: 0.7877 - val_acc: 0.8849
Epoch 3/64
```

```
2019-03-01 13:09:14.254 CET master-replica-0 - 15s - loss: 0.6651 - acc: 0.7797 - val_loss: 1.0527 - val_acc: 0.9352
2019-03-01 13:09:14.253 CET master-replica-0 Epoch 3/64
2019-03-01 13:08:58.250 CET master-replica-0 - 15s - loss: 0.7169 - acc: 0.7470 - val_loss: 1.1322 - val_acc: 0.9452
2019-03-01 13:08:58.250 CET master-replica-0 Epoch 2/64
2019-03-01 13:08:41.771 CET master-replica-0 - 20s - loss: 0.9439 - acc: 0.5947 - val_loss: 1.0670 - val_acc: 0.7330
2019-03-01 13:08:41.771 CET master-replica-0 Epoch 1/64
2019-03-01 13:08:41.771 CET master-replica-0 Train on 283563 samples, validate on 50041 samples
```

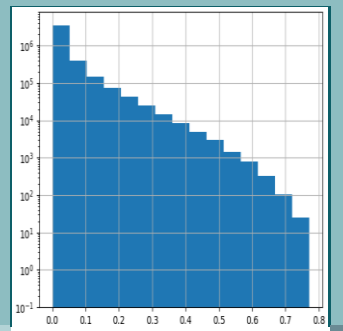
# Quelques vrais chiffres

## de notre algorithme en production.

**x n**

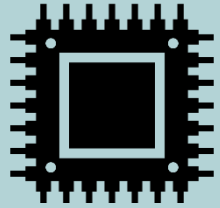
**150Gb**
données prétraitées

**1.5M**
prédictions quotidiennes



**450M**
événements

**100M**
séquences d'entraînement



# Et maintenant,
## quels leviers d'actions marketing ?

**Madame X a une probabilité de 4/10 d'acheter un pantalon dans les 12 jours.**

**Ciblage marketing dans le magasin favori.**

**Monsieur Y a une probabilité de 2/10 chances d'acheter un tee-shirt.**

**Ciblage marketing Promo sur canal sms.**

**Madame Z a une probabilité de 1/10 d'acheter.**

**Aucune action marketing.**

Et ce n'est que le début **d'un long processus itératif.**

# NOS RÉFÉRENCES


**Parce que ce sont nos clients qui parlent le mieux de nous.**

Think Data. Think Ysance.

# Merci !

## Avez-vous des questions ?

# Stand A42

Thomas Gorbinet
**Lead Data Scientist**
**thomas.gorbinet@ysance.com**

Aurélien Bénard
**Lead Data Scientist**
**aurelien.benard@ysance.com**

Ysance