CX, MODEL TRAINING AUTOMATION AND THE FEATURE STORE PROBLEM

September 26, 2019
WHAT WE WILL TALK ABOUT TODAY

• ML USE CASE - CUSTOMER EXPERIENCE
• DATA AT SCALE IS HARD
• WHY WE NEED A FEATURE STORE
• SOLVING THE FEATURE STORE PROBLEM
A global media and technology company with several businesses, including Comcast, NBCUniversal, and Sky.

<table>
<thead>
<tr>
<th>COMCAST</th>
<th>Products &amp; Services</th>
<th>Cable Networks</th>
<th>Broadcast</th>
<th>Film</th>
<th>Parks</th>
<th>Products &amp; Services</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comcast Spectacor</strong></td>
<td>Competition</td>
<td></td>
<td>Competition</td>
<td></td>
<td></td>
<td>Competition</td>
</tr>
</tbody>
</table>

(Format note: The table structure is not filled with any data. The table is meant to represent a structure for organizing information.)
ML Lifecycle – Roles & Workflow

Inception
- Define Use Case

Exploration
- Identify Model Types & Explore Features
- Create and publish new features

Model Development
- Create & Validate Models

Candidate Model Selection
- Model Review
- Model Selection

Model Operationalization
- Collect new data & retrain
- Define Online Feature Assembly
- Define pipeline to collect outcomes
- Model Deployment and Monitoring

Model Evaluation
- Evaluate Live Model Performance
- Go Live with Selected Models

Go Live Phase
- Monitor Live Models
What is the Business Problem?

• We have Inefficient Processes == WASTE
• The WASTE costs money
• We don’t use all of the data we have to provide an ideal customer experience

• GOAL: Take cost out of the business + drive KPIs
  • IVR / Digital Containment
  • XA / Chat Containment
  • Rework
  • Avoidable Truck Rolls
Machine Learning and Human Empathy

Six Million Dollar Man

• The Six Million Dollar Man is a better analogy
  • Start with Quality People
  • Augmenting them with Technology
  • Up-skilling the Workforce

• Machine Learning + Human Decision Making = Better Results
Operational Waste During Service Calls
Xfinity Assistant
Web, Product and Mobile
ML Powered Service Calls
Machine Learning at Scale

STREAMING EVENTS
Product supply chain produces billions of features are engineered and enriched to create the ML dataset

RECORDS UPDATES
Multiple feature sets are made available in the online feature store

CUSTOMER ACCOUNTS
Predictions are ready to be made for Comcast customers that use Internet and Video products

BILLIONS AND BILLIONS
HUNDREDS OF MILLION
TENS OF MILLION
Customer Impacting Predictions YTD

- Millions of Predictions
- 60% HSD Predictions
- 40% X1 Predictions

85% of customers that used the recommendations did not call for repair in a 24-hour period
TECHNICAL OVERVIEW

- METADATA
- DATA INGESTION
- FEATURE ENGINEERING
  - ”Feature store problem”
- MODEL TRAINING
- MODEL DEPLOYMENT

- MODEL SERVING
- AUTOMATION
- BRINGING IT ALL TOGETHER
START SIMPLE - ABSTRACT

DATA INGESTION AND FEATURE ENGINEERING

HOW TO MANIPULATE AND STORE DATA?
ABSTRACT TO CONCRETE

DATA INGESTION AND FEATURE ENGINEERING

<table>
<thead>
<tr>
<th>input</th>
<th>f</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$f$</td>
<td>$f(x)$</td>
</tr>
</tbody>
</table>

- Kafka
- Kinesis
- HTTP requests
- batch, files, etc.

- Windowing
- Aggregation
- Normalization
- Transformation
- etc.

Features ready to be passed into ML models
WHAT IS THE FEATURE STORE PROBLEM?

• WHAT IS A FEATURE STORE
• WHY DO YOU NEED ONE
• WHAT IS THE PROBLEM?
FEATURE STORE

TYPES OF FEATURE STATES TO PERSIST

• WINDOWED DATA
• AGGREGATE DATA
• OTHER DATA RELATED TO INTERMEDIATE STEPS FOR FEATURE ENGINEERING
  • Common indexes
  • Inference and predictions from other models
  • Etc.

Can be many things:

• Spark state store (Cassandra, RocksDB)
• Flink state backend (Memory, RocksDB)
• Custom: Hadoop (HDFS, HBase)
• Custom: Redis
RESEARCH AND MODEL FLOW

MODEL TRAINING AND DEPLOYMENT
# Start a MLFlow experiment and label it
with mlflow.start_run(run_name="no outliers, default hyperparams"):  
    # train
    clf = train(train_x, train_y, solver, C, multi_class)  
    # predict
    predict = clf.predict(test_x)
    # eval metrics
    rmse, mae, r2 = eval_metrics(test_y, predict)

    # log some params
    mlflow.log_param("c", C)
    mlflow.log_param("multi_class", multi_class)
    # log some metrics
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("r2", r2)
    mlflow.log_metric("mae", mae)

    # finally log the model and end
    mlflow.sklearn.log_model(clf, "model")
    mlflow.end_run()
MLFLOW

MODEL EXPERIMENT TRACKING

![MLflow Experiment Tracking Interface](image)
import pickle

class CalifHousingPredictor(object):

    def __init__(self):
        # load the saved trained model
        model_file = 'linear_model.pkl'
        self.sklearn_model = pickle.load(open(model_file, 'rb'))

    def predict(self, X, features_names):
        return self.sklearn_model.predict(X)
MODEL SERVING - MANY POSSIBILITIES

import pickle

class CalifHousingPredictor(object):
    def __init__(self):
        # load the saved trained model
        model_file = 'linear_model.pkl'
        self.sklearn_model = pickle.load(open(model_file, 'rb'))

    def predict(self, X, features_names):
        return self.sklearn_model.predict(X)

kubeflow HTTP endpoint (seldon)

as part of a chain for ensembles

directly invoke
PIPELINE AUTOMATION

1. Researchers request a new feature source.
2. Feature engineering controller retrieves metadata from Apache Atlas.
3. Discovery and metadata create a multi-step workflow in Argo.
4. 1. Build stream object deserializers.
5. 2. Build Docker image.
6. 3. Push image to Docker hub.
7. 4. Generate deployment.
8. 5. Deploy.
BRINGING IT ALL TOGETHER

MODEL SERVING WITH SEPARATE FEATURE ENGINEERING
PERFORMANCE PIPELINE
(SOLVING THE FEATURE STORE PROBLEM)

MODEL SERVING WITH INTEGRATED FEATURE ENGINEERING
BUT IN-MEMORY IS SMALL?

MODEL SERVING: SPLIT UP THE IN-MEMORY DB BY FUNCTION
DEMO CODE

EXAMPLE CODE USING KUBEFLOW, KAFKA, PYTHON, REDIS AND A SIMPLE MODEL

HTTPS://GITHUB.COM/COMCASTSAMPLES

“END TO END ML FEATURE STREAMING WITH KUBEFLOW KAFKA AND REDIS”