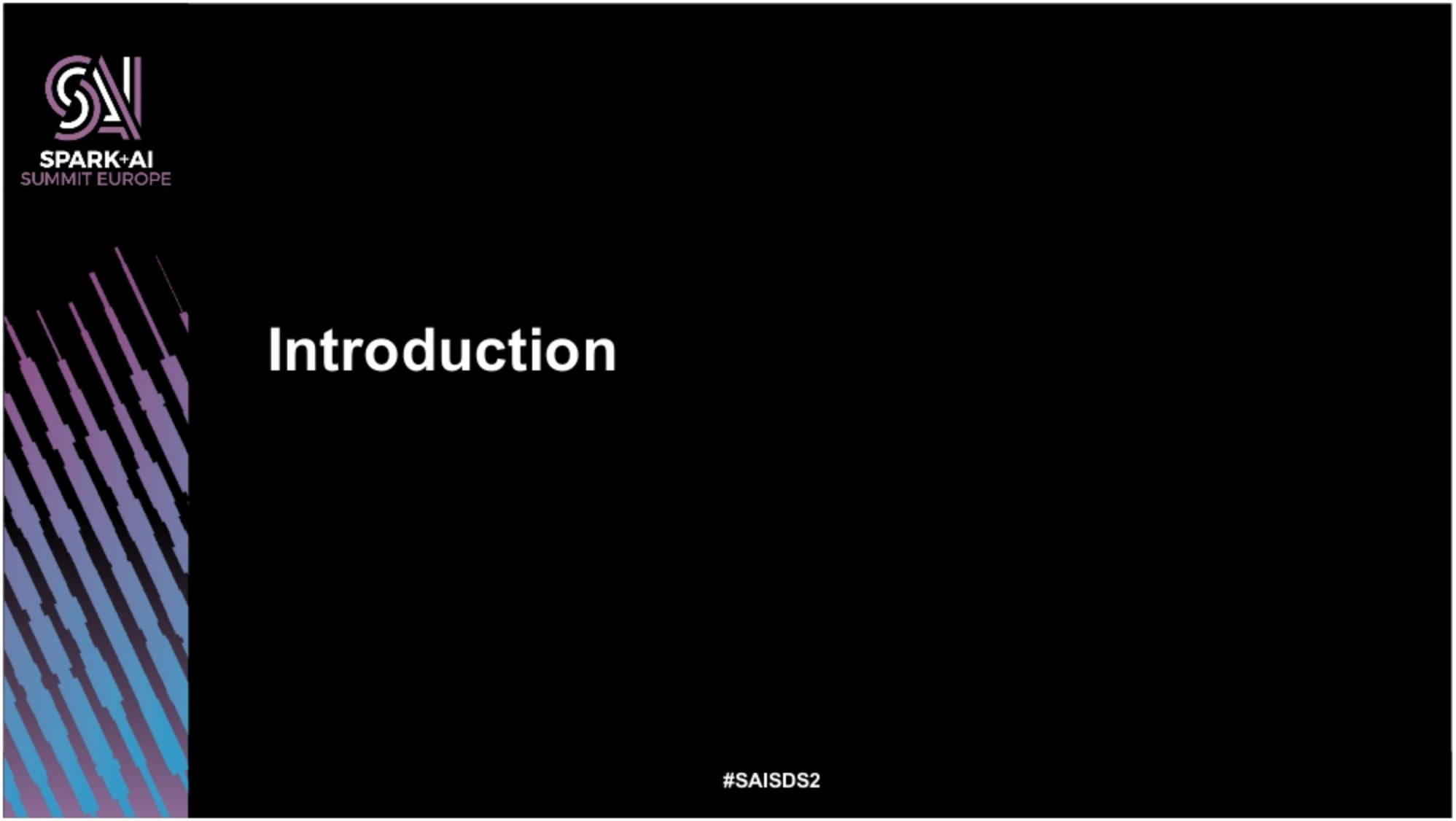
Deploying Python Machine Learning Models with Apache Spark

Brandon Hamric & Alex Meyer, Eventbrite

#SAISDS2


SPARK+AI
SUMMIT EUROPE
Introduction
#SAISDS2

About Eventbrite

- Global ticketing and event technology platform that provides creators of events of all shapes and sizes with tools and resources to seamlessly plan, promote, and produce live experiences around the world
- Can be accessed online or via mobile apps, scales from basic registration and ticketing to a fully featured event management platform
- 203 million tickets processed in 2017
- Powered 3 million events in 170+ countries in 2017
- 700k creators supported in 2017

About Us

- Eventbrite
- We're data engineers
- We ship models for Eventbrite data scientists
- Started out at Eventbrite on Discovery - Event Recommendations
- Built new data infrastructure to support all business needs
- Our team creates, maintains, and supports the data infrastructure, tasks, and pipelines that serve other engineers, business insights, and product

Brandon Hamric - bhamric@eventbrite.com

- Principal Data Engineer/Architect @ Eventbrite
- Co-founded Rescue Forensics (YC W15)
- 10 years experience in data engineering
- Worked with Spark since 2014

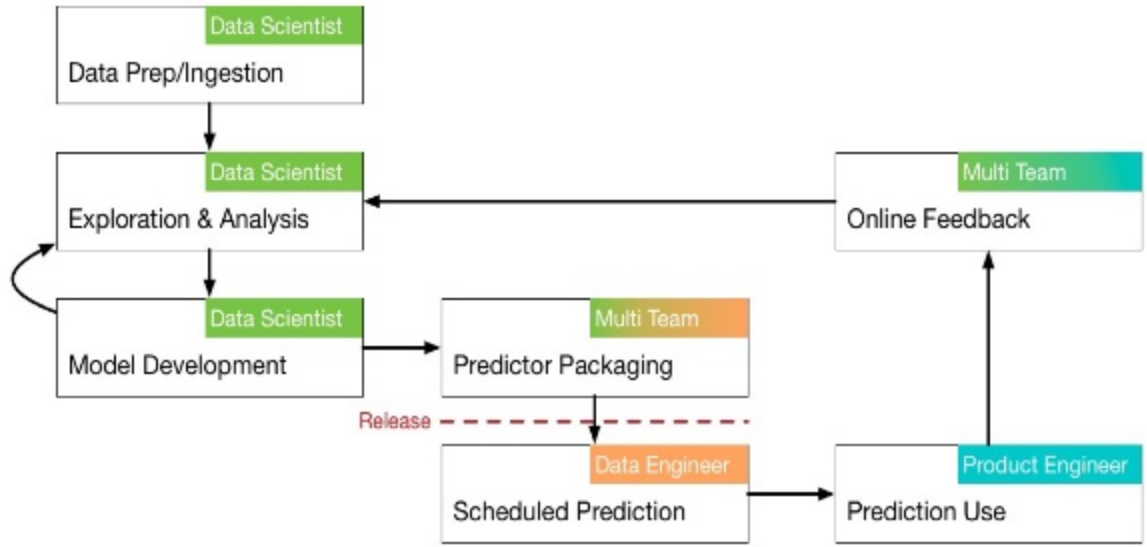
Alex Meyer - alexm@eventbrite.com

- Senior Data Engineer @ Eventbrite
- MS in Computer Science - Distributed Systems (Vanderbilt University)
- 4 years experience in data engineering
- Worked with Spark since 2014

Structured Predictors

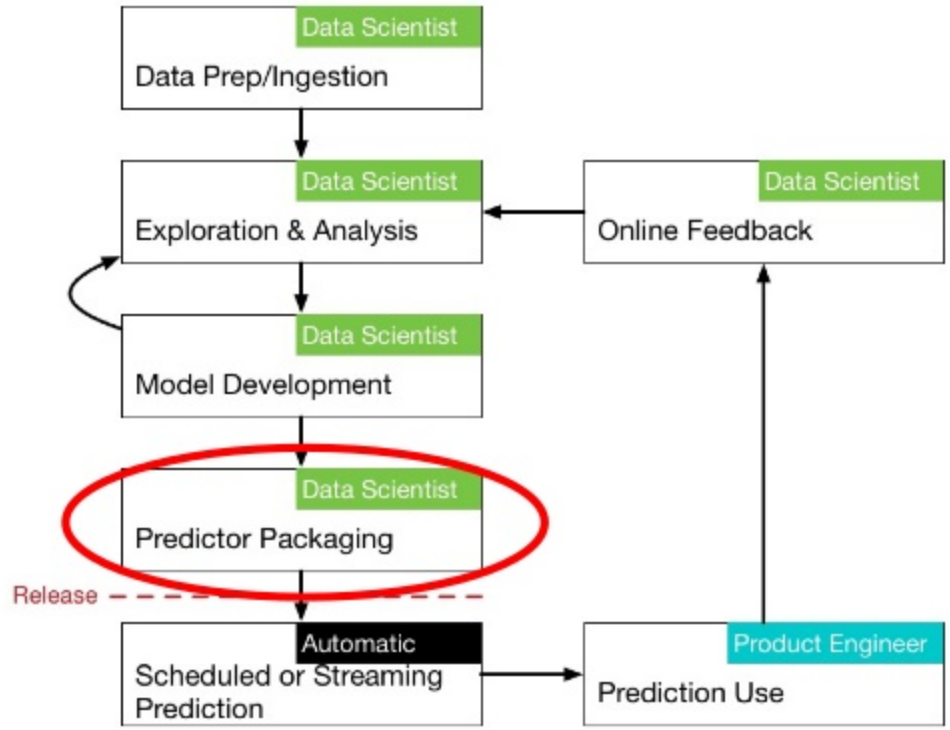
Common Predictor Workflow

- High coupling between engineers and data scientists
- Mostly serial workflow
- High barrier to entry
- Too many contributors
- Code duplication



Improved Predictor Workflow

- Low coupling between engineers and data scientists
- Independent Workflows
- Data scientists own their models end-to-end
- Data Engineering isn't a bottleneck



Predictor Code

- Training and prediction code can be inconsistent
- Sample data prep can be different than prod data prep

- Training and prediction code can be inconsistent
- Mostly written for vertical scaling

- Version management is hard
- Can use a lot of memory
- It can be hard to switch between models

- Bulk vs single-item

Predictor Deployment Problems

- Shared code between dev, batch, and streaming is an afterthought
- Most models are written for vertical scaling first
- Deployment is ad-hoc without a common structure
- Model iteration is slow because of lack of automation
- Model versioning isn't consistent without a library

Predictor Structure

	Notebook	Offline Prediction	Streaming Prediction
Data prep and cleanup	Query to a local csv	Convert to incremental query	Convert to read stream
Feature Extraction	Pandas dataframes and python functions	Convert to spark dataframe or rdd operations	Convert to dataframe operations or foreachBatch in Spark 2.4
Load Model	Load from a local pickle	Load from s3 or hdfs onto executors	Load from s3 or hdfs onto executors
Predict	Mixed into scoring logic	Mapper or UDF on features	UDF on feature rows

Predictor Class

- Manages Model
  - Versioning
  - Storage
  - Loading
- Outlines structure
  - Data loading
  - Feature extraction
  - Prediction
- Batch and streaming
- Enables Automation

```python
class Predictor(object):
    @property
    def name(self):
        return self.__class__.__name__

    @staticmethod
    def save_model(model, model_key, metadata=None):
        s3_put(Bucket=s3_bucket, Key=self.name, metadata=metadata)

    def load_model(self, version='latest'):
        self.model = s3_get(Bucket=s3_bucket, Key=self.name, Version=version)

    def list_versions(self):
        s3_list_versions(Bucket=s3_bucket, Key=self.name)

    def load_input_stream(self, spark):
        raise NotImplemented

    def load_input_dataframe(self, spark):
        raise NotImplemented

    def extract_features_bulk(self, rows):
        for row in rows:
            yield self.extract_features(row)

    def extract_features(self, row):
        raise NotImplemented

    def predict_bulk(self, feature_rows):
        for feature in feature_rows:
            yield self.predict_item(feature)

    def predict_item(self, features):
        raise NotImplemented
```

Example Predictor and Demo

Demo - Latent Dirichlet Allocation (LDA)

- Generate topics on Eventbrite's event description corpus
- Get topic probabilities per event
- We can use topics to improve search, browse, and personalization
- LDA Wiki
- LDA Scikit Learn Model

<open notebook>

Takeaways

- Consistent predictor structure makes distributed prediction easy to automate deployment
- Streaming and batch prediction can share code
- Use bulk feature extraction and prediction often
- We may opensource our predictor library
- We're hiring!

Questions? Feel free to reach out!

Thanks!