From Whiteboard to Production System

Demand Forecasting System for an Online Grocery Retailer

Robert Pesch & Robin Senge

Strata Data Conference NYC, September 2019
Rewe and inovex drive big data and data science initiatives in order to optimize supply chain processes since 2015
Customers of the Rewe delivery service order grocery for future delivery

1. Go to the online shop or mobile app
2. Fill your basket
3. Select a future delivery slot
4. Shop checks availability
5. Receive your purchase
Mastering the supply chain is even more important to the Rewe delivery service than to a regular food retailer.

Availability of products play a key role to success of the business case.
An improved demand forecasting system for e-Grocery has huge potential for increasing the availability of articles.

Reasons for unavailability (several possible per case):
- Not available articles: 100.0%
- Inaccurate predictions: 80%
- Central logistic problems: 20%
- Unexpected inventory correction: 17%
- Unexpected spoilage: 7%
Setup
Data science projects face additional challenges compared to regular software development projects

- Complexity of the task (at least ‘complex’ in terms of Cynefin framework)
- High uncertainty about usefulness of models
- You cannot plan your success
- Wrong team setup
An iterative approach is even more important to success in data science products than traditional software.
Create a feeling of joint responsibility – Create a common goal

Supply Chain Process Owner

Data Scientist

Software Engineer

Big Data Platform Engineer

\[ y = X\beta + \varepsilon \]
KEEP CALM AND TRUST YOUR TEAM
On the Whiteboard
We measure the real customer request during checkout
Interactive data exploration and evaluation reports help understanding your data

- Learn more about your data
- Evaluate current solution and refined models
- Generate new ideas
There will not be a single best model for all products

- Demand patterns vary strongly among products
- Long tail: high and low demand products
- Following the „No free lunch“-theorem there will not be a single best model
Evaluate simple models to test your pipelines and to learn more about your data

- Last observed data point
- Moving average
- Constant value, ...
Classical time-series methods are well researched and statistical sound methods

**Autoregressive Model**

\[ X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \]

**Exponential Smoothing**

\[ y_t^* = \alpha \cdot y_t + (1 - \alpha) \cdot y_{t-1}^*. \]

**(S)ARIMA(X), Prophet, ...**

...
The ability to integrate exogenous variables is limited
Regression models enable the utilisation of arbitrary features and algorithms.

- Known orders
- Calendar information
- Price
- Promotions
- Demand averages
- Geographical location, ...

### Transform and define features

<table>
<thead>
<tr>
<th>ID</th>
<th>Date</th>
<th>Price</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>2017-03-31</td>
<td>2.69</td>
<td>114</td>
</tr>
<tr>
<td>00002</td>
<td>2017-03-31</td>
<td>0.49</td>
<td>111</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>99999</td>
<td>2018-03-31</td>
<td>1.79</td>
<td>121</td>
</tr>
</tbody>
</table>

$$y = X\beta + \varepsilon$$
Each model class comes with strengths and weaknesses

**Nearest Neighbours**
- Easy to interpret
- Limited extrapolation capacity

**Linear Regression**
- Easy to interpret
- Limited expressiveness (linear dependencies)

**Boosted Regression Trees**
- Already strong out-of-the-box
- Not much data preparation necessary
- Limited extrapolation capacity
- Prone to outliers “blind spots”

**Artificial Neural Networks**
- Potentially strong model class
- Specialized topologies (LSTM)
- Need lots of computing power
- High effort in engineering

**Ensembles**
- Ensemble effect helps combining the strengths of different models
- High effort to support many models
Automated model selection enables choosing one model per product

- Finding the best models manually does not scale
- Automated mapping of best models using error metrics
- Enable adaption to demand pattern changes
- Legacy system as benchmark and fallback model

Towards Production System
Machine learning code is only a tiny part of a data science product
The forecasting system comprises many components:

- **Data collector**: Collects raw data.
- **Feature Generator**: Generates cleaned data from raw data.
- **Prediction Workflow**: Generates features from cleaned data.
- **Outlier Detection**: Identifies outlying predictions.
- **Blacklist & Importer**: Imports results and filtered predictions.

**Runs once per day**:
- **Trainer & Evaluator**: Trains and evaluates models.
- **Model Selector**: Selects the best model.
- **Results**: Outputs results for each run.
- **Simulator**: Simulates KPI simulations.

**Runs once per month**:
- **Blacklist**: Stores blacklisted components.

**Complexity**:
- Ranges from low to high complexity.
The technology stack

<table>
<thead>
<tr>
<th>Category</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend and Monitoring</td>
<td>🐸 jupyter, 🎈 Grafana, 🔥 Prometheus</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>🧪 Python, 🔴 R, 🎨 Python</td>
</tr>
<tr>
<td>Languages</td>
<td>🍒 Java, 🐢 R, 🎨 Python</td>
</tr>
<tr>
<td>Data Processing Frameworks</td>
<td>🍀 Apache Spark, 🍀 Apache Kafka, 🍀 PySpark</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>🍀 Google Cloud Platform, ⛓ Compute Engine, 🍀 Cloud Dataproc, 🍀 Hadoop</td>
</tr>
</tbody>
</table>
PySpark enables us to parallelize training and application of scikit-learn models

- Many models
- Parallelize model training and evaluation
- Prototyped models can be used directly

```python
from sklearn.linear_model import Lasso

def train(key_value):
    article = key_value[0]
    features = key_value[1]
    # ...
    model = Lasso()
    model.fit(...)
    # ...

    features = spark.read.parquet(...)

    rdd = features.
    map(lambda row: (row._2,row._1)).
    groupByKey().
    map(train).
```
Monitoring
Monitoring is an integral part of all productive software systems

- Monitoring your system components for errors and failures is the standard

Add the following:

- Check your input data for validity
- Check your output data for plausibility
Can I trust my input data?

- Do outlier detection!

- Apply e.g.
  - quantile filters
  - rules
  - proximity based methods

- Act on them by
  - imputation
  - removal of data points
  - skip features
Automated checks test millions of predictions and lets you focus

- Manual checking of millions of predictions is not feasible

- Predictions might suffer from:
  - yet hidden programming bugs
  - instable models
  - broken assumptions
  - blind spots

- Focus on predictions that seem strange
Results
New system reduces non-availability by 50% compared to legacy system
Forecasting the whole demand distribution enables us to manage the trade-off between availability and spoilage

- Most models predict expected value assuming symmetric costs
- Actually e-grocery exhibits asymmetrical costs and a non-trivial cost-function
- Specific for each individual product (esp. spoilage)

*Distributional Regression for Demand Forecasting in e-Grocery* - https://ssrn.com/abstract=3312609
Take Home Messages

• Reduce complexity iteratively by an agile process

• Create a feeling of joint responsibility in your team

• There is no free lunch and recall Occam’s razor

• Automate input and output outlier checks

• Keep it simple as long as possible, it gets complex early enough
Vielen Dank

Dr. Robert Pesch
Dr. Robin Senge

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe
GERMANY

robert.pesch@inovex.de
robin.senge@inovex.de