Deep Learning with Structured Data

February 11, 2021

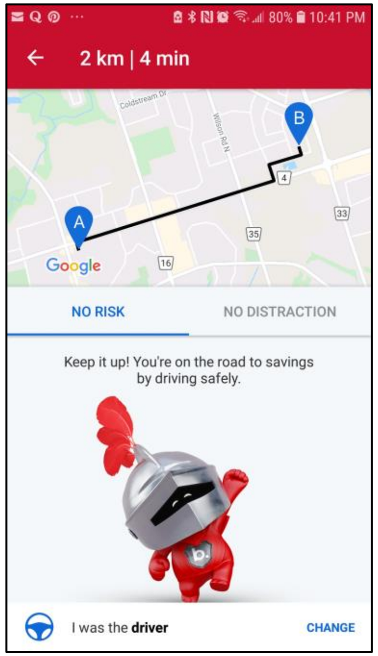
Mark Ryan

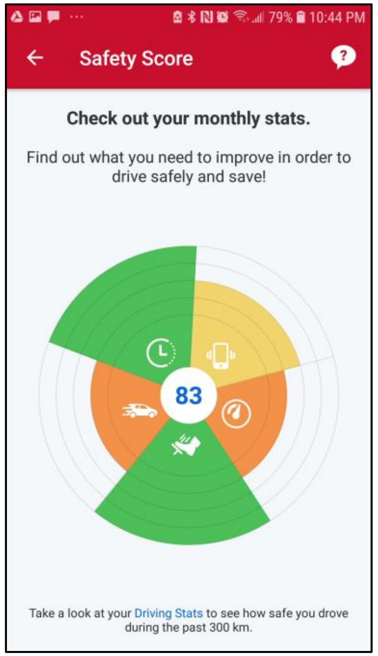
Agenda

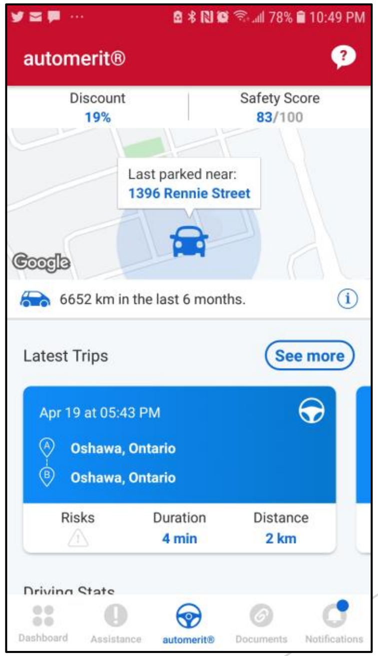
- Background
- Why use deep learning for problems involving tabular, structured data?
- Walk through the end-to-end approach:
  - Data cleanup
  - Building & training the deep learning model (including bakeoff with XGBoost)
  - Deployment
- Potential future enhancements
- Resources for learning more on the topic

Background

- Computer Science at the University of Toronto in the golden age of GOFAI
- Since Oct 2019, Data Science Manager in the Data Lab at Intact Insurance
- Additional interests: applications of GPT-3, chatbots, self-driving vehicles







What Is Structured Data?

- For the purposes of this discussion, **structured data** is tabular data organized in rows and columns
- Contrast with non-tabular data:
  - Images
  - Audio
  - Free-form text
- This kind of data has a structure, but is not tabular
- By this definition, structured data includes tables with columns containing unstructured data, such as free-form text

Why Deep Learning with Structured Data?

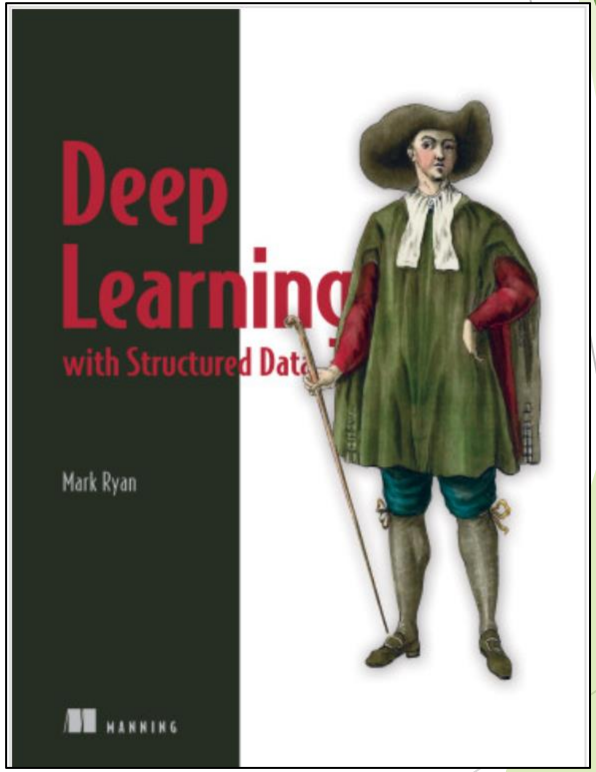
- Deep learning is the rocket fuel of machine learning
- Introductory deep learning examples have nothing to do with everyday jobs
- People want to learn about deep learning, but their jobs are about tables, not recognizing pictures of cats



Images: Pixabay

Deep Learning with Structured Data: Genesis of the Book

- Lack of examples of deep learning applied to problems I cared about
- Exercised a simple deep learning model on problems in the Db2 support lead role:
  - Predicting time to resolution of tickets
  - Predicting duty manager calls
- Blogs on Medium caught Manning Publication’s attention
- Book is available at Manning and Amazon



Deep Learning with Structured Data: What Are the Goals of the Book?

- Make an argument for deep learning *as an option* for solving problems involving structured data
- Show a simple, end-to-end solution built around a deep learning model, featuring:
    1. A real-world structured dataset
    2. An accessible but complete stack:
        1. Pandas for representing tables in Python
        2. Keras functional API for deep learning framework – on top of TensorFlow 2
        3. Scikit-learn for pipelines
        4. Flask / Facebook Messenger + Rasa for deployment
    3. Useful coding ideas:
        1. config files
        2. logging
        3. Keras callbacks

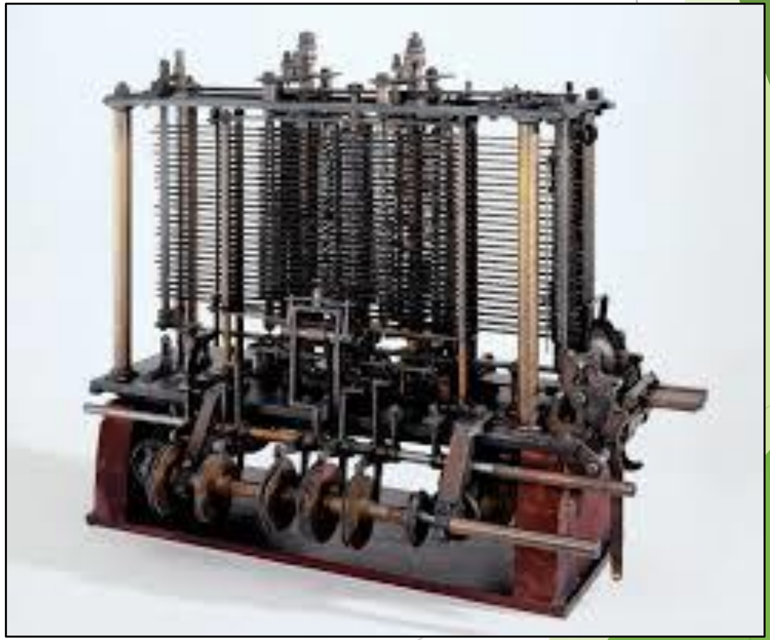
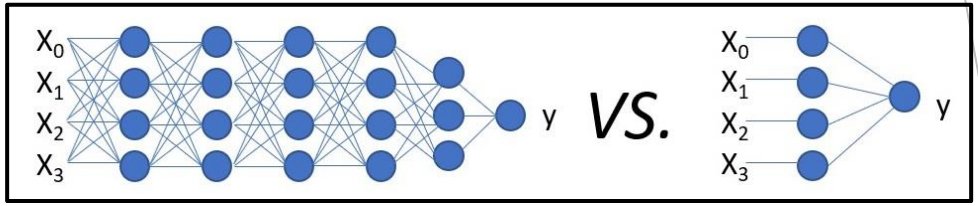


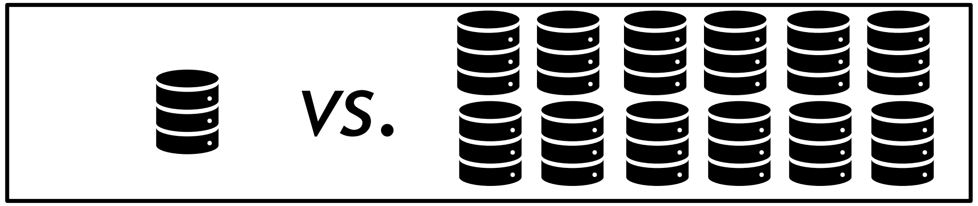
Image: brittanica.com

Objections to Deep Learning with Structured Data

- Deep learning is more complicated



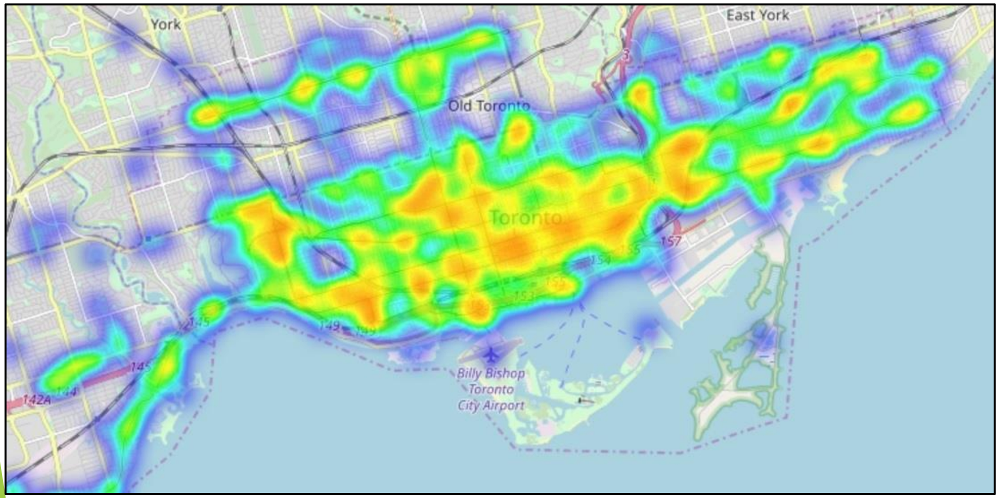
- Structured datasets are too small



- XGBoost wins Kaggle competitions - why mess with success?

A Problem to Tackle – Streetcar Delays

- Couldn’t use IBM datasets from earlier deep learning experiments
- Found a [publically available streetcar delay dataset]
- Train a model on this dataset to **predict whether a given streetcar trip would be delayed**



A Real-World Dataset

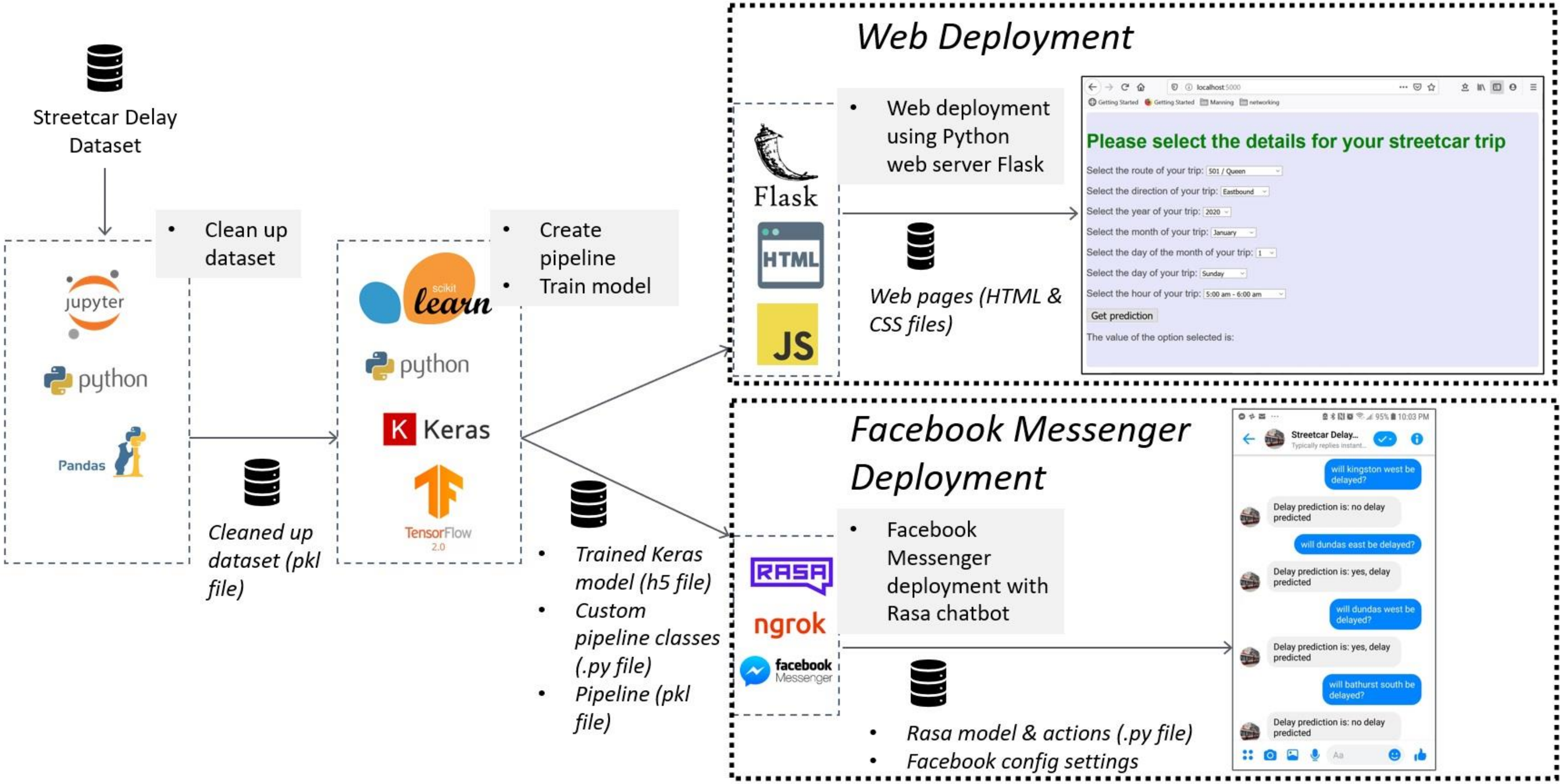
- ~80 K records – all streetcar delays since Jan. 2014
- An XLS file / year; one tab / month
- Very messy

Report Date	Route	Time	Day	Location	Incident	Min Delay	Min Gap	Direction	Vehicle
2014-12-17	504	9:24:00 AM	Wednesday	Dundas West Stn	Mechanical	34	38	w	4055
2014-12-18	506	2:55:00 PM	Thursday	RUSSELL YARD	Mechanical	5	10	eb	4152
2014-12-19	505	10:08:00 AM	Friday	King and Shaw	Investigation	2	5	sw	4248

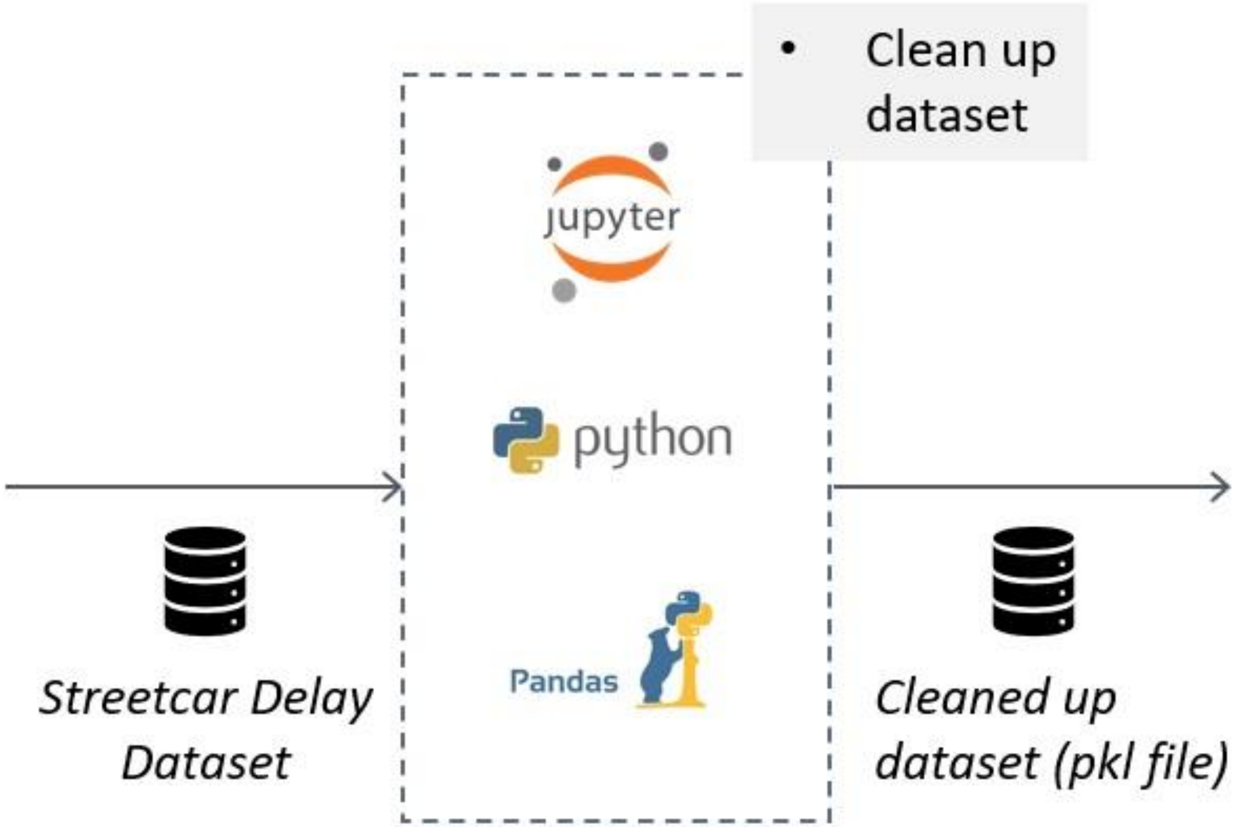
Report Date	Route	Time	Day	Location	Incident	Min Delay	Min Gap	Direction	Vehicle
01-Jul-18	301	12:06:00 AM	Sunday	Neville park	Held By	244	253	B/W	4030
01-Jul-18	301	4:05:00 AM	Sunday	Long branch loop	Mechanical	30	60	E/B	4165
01-Jul-18	501	6:03:00 AM	Sunday	Russell Yard	Late Leaving Garage	9	18	E/B	4067

Report Date	Route	Time	Day	Location	Incident ID	Incident	Delay	Gap	Direction	Vehicle
01-Apr-19	512	4:26:00 AM	Monday	Roncesvalles Yard.	1	Mechanical	10	20	E/B	4460
01-Apr-19	501	4:27:00 AM	Monday	Queen St. E and Woodfield Ave.	1	Mechanical	17	17	E/B	4189
01-Apr-19	501	4:37:00 AM	Monday	Queen St. E at Greenwood Ave.	1	Mechanical	5	10	W/B	4012

Accessible but Complete Stack



Clean Up the Data



Clean Up the Data

Report Date Time	Report Date	Route	Time	Day	Location	Incident	Min Delay	Min Gap	Direction	Vehicle	Report Date Time	year	month	daym	hour	time_of_day
2016-01-01 00:00:00	2016-01-01	505	00:00:00	Friday	dundas west stationt to broadview station	General Delay	7.0	14.0	w	4028	2016-01-01 00:00:00	2016	1	1	0	overnight
2016-01-01 02:14:00	2016-01-01	511	02:14:00	Friday	fleet st. and strachan	Mechanical	10.0	20.0	e	4018	2016-01-01 02:14:00	2016	1	1	2	overnight
2016-01-01 02:22:00	2016-01-01	301	02:22:00	Friday	queen st. west and roncesvalles	Mechanical	9.0	18.0	w	4201	2016-01-01 02:22:00	2016	1	1	2	overnight
2016-01-01 03:28:00	2016-01-01	301	03:28:00	Friday	lake shore blvd. and superior st.	Mechanical	20.0	40.0	e	4251	2016-01-01 03:28:00	2016	1	1	3	overnight
2016-01-01 14:28:00	2016-01-01	501	14:28:00	Friday	roncesvalles to neville park	Mechanical	6.0	12.0	e	4242	2016-01-01 14:28:00	2016	1	1	14	midday

	Report Date	count	Route	Direction	hour	year	month	daym	day	Min Delay	target
0	2014-01-01	0	301	e	0	2014	1	1	2	0.0	0
1	2014-01-01	0	301	e	1	2014	1	1	2	0.0	0
2	2014-01-01	0	301	e	2	2014	1	1	2	0.0	0
3	2014-01-01	0	301	e	3	2014	1	1	2	0.0	0
4	2014-01-01	0	301	e	4	2014	1	1	2	0.0	0

Build and Train Model & Pipeline

- Create pipeline
- Train model

scikit learn

python

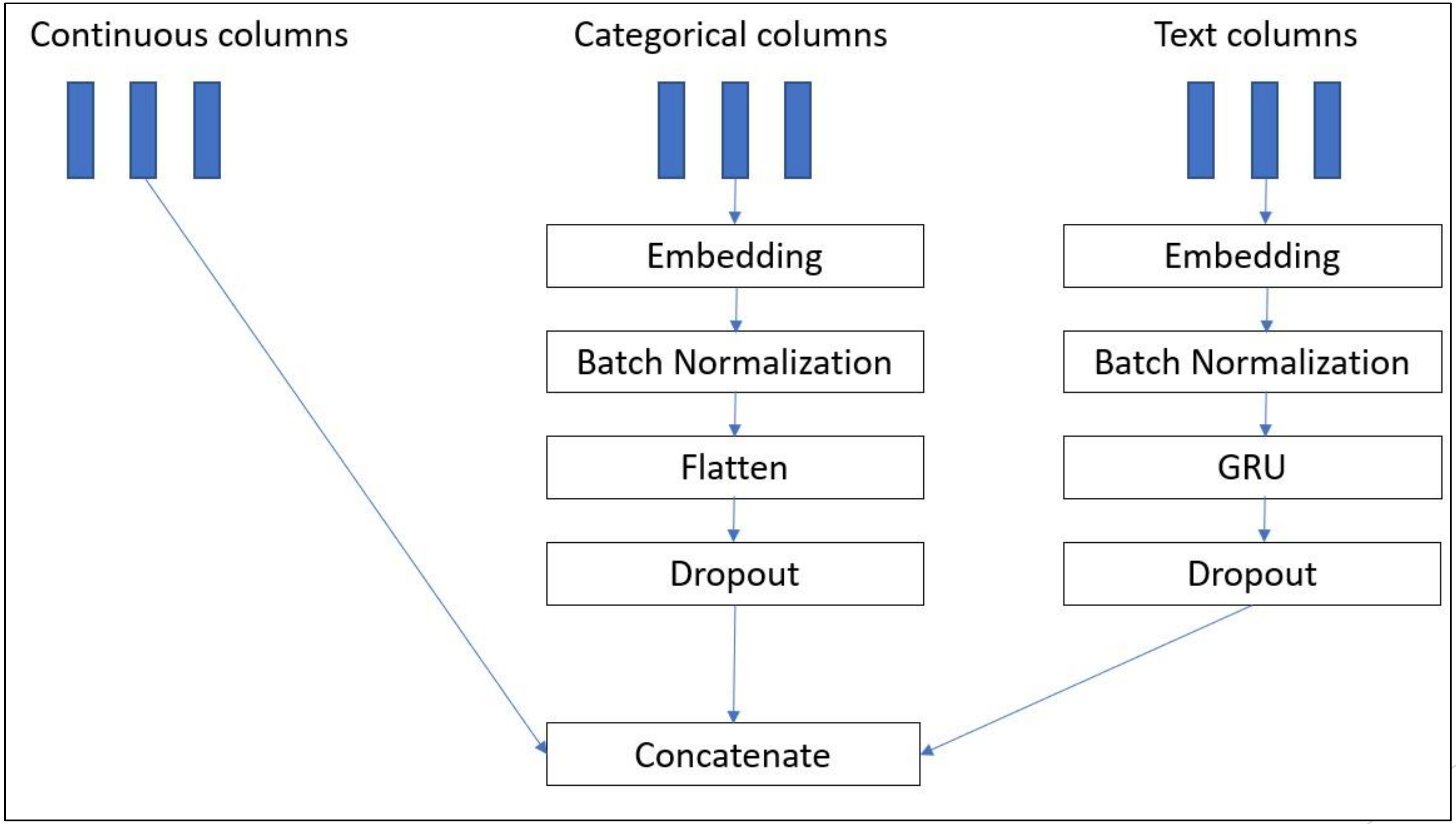
Keras

TensorFlow 2.0

Cleaned up dataset (pkl file)

- *Trained Keras model (h5 file)*
- *Custom pipeline classes (.py file)*
- *Pipeline (pkl file)*

Build Model: Keras Model Layers



Build Model: Code that Generates the Keras Model using Functional API

1

```
for col in collist:
    catinputs[col] = Input(shape=[1],name=col)
    inputlayerlist.append(catinputs[col])
    embeddings[col] = (Embedding(max_dict[col],catemb) (catinputs[col]))
    # batchnorm all
    embeddings[col] = (BatchNormalization() (embeddings[col]))
    collistfix.append(embeddings[col])
```

2

```
# define layers for text columns
if includetext:
    for col in textcols:
        print("col",col)
        textinputs[col] = Input(shape=[X_train[col].shape[1]], name=col)
        print("text input shape",X_train[col].shape[1])
        inputlayerlist.append(textinputs[col])
        textembeddings[col] = (Embedding(textmax,textemb) (textinputs[col]))
        textembeddings[col] = (BatchNormalization() (textembeddings[col]))
        textembeddings[col] = Dropout(dropout_rate) ( GRU(16,kernel_regularizer=l2(l2_lambda)) (textembeddings[col]))
        collistfix.append(textembeddings[col])
        print("max in the midst",np.max([np.max(train[col].max()), np.max(test[col].max())])+10)
    print("through loops for cols")
```

3

```
# define layers for continuous columns
for col in continuouscols:
    continputs[col] = Input(shape=[1],name=col)
    inputlayerlist.append(continputs[col])
```

Train Pipeline

Raw input:

Report Date	Route	Time	Day	Location	Incident	Min Delay	Min Gap	Direction	Vehicle
2014-01-02	505	6:31:00 AM	Thursday	Dundas and Roncesvalles	Late Leaving Garage	4	8	E/B	4018
2014-01-02	504	12:43:00 PM	Thursday	King and Shaw	Utilized Off Route	20	22	E/B	4128
2014-01-02	501	2:01:00 PM	Thursday	Kingston road and Bingham	Held By	13	19	W/B	4016
2014-01-02	504	2:22:00 PM	Thursday	King St. and Roncesvalles Ave.	Investigation	7	11	W/B	4175
2014-01-02	504	4:42:00 PM	Thursday	King and Bathurst	Utilized Off Route	3	6	E/B	4080

python

Cleaned up and refactored:

Report Date	count	Route	Direction	hour	year	month	daym	day	Min Delay	target
2014-01-01	0	301	e	0	2014	1	1	2	0.0	0
2014-01-01	0	301	e	1	2014	1	1	2	0.0	0
2014-01-01	0	301	e	2	2014	1	1	2	0.0	0
2014-01-01	0	301	e	3	2014	1	1	2	0.0	0
2014-01-01	0	301	e	4	2014	1	1	2	0.0	0

1 Encode categorical values

2 Convert dataframe to list of np arrays

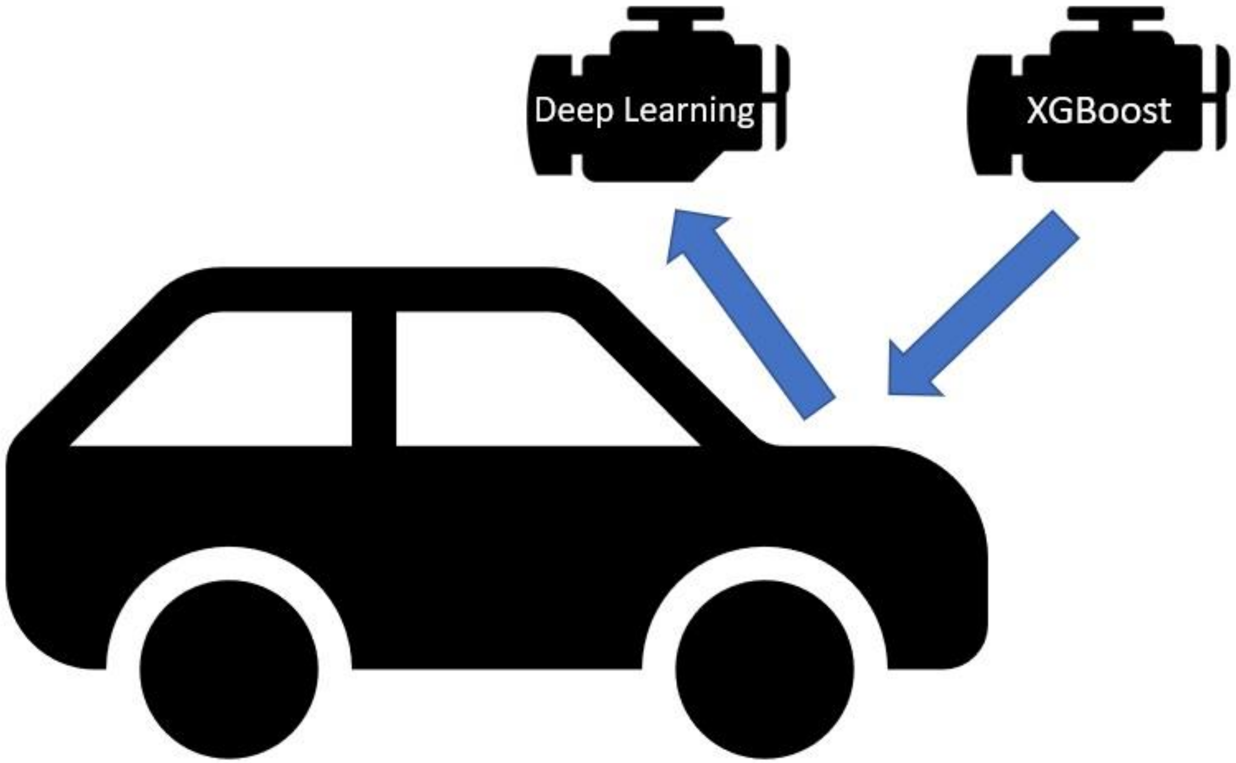
What the model expects:

- Hour: 18
- Route: 0
- Day of the month: 21
- Month: 0
- Year: 5
- Direction: 1
- Day of the week: 1

Results of a Set of Training Experiments

Experiment	Epochs	Early stop enabled?	Weight for “1” (delay) values	Early stop controls		Terminal Validation accuracy	False negatives exercising model on test set	Recall on test set: true positive / (true positive + false negative)
				monitor	mode			
1	10	no	1.0	NA	NA	0.98	11,000	0
2	50	no	1.0	NA	NA	0.75	7,700	0.31
3	50	no	No delay / delay	NA	NA	0.8	4,600	0.59
4	50	yes	No delay / delay	Validation loss	min	0.69	2,600	0.76
5	50	yes	No delay / delay	Validation accuracy	max	0.72	2,300	0.79

Deep Learning vs. XGBoost



Deep Learning vs. XGBoost

Category	XGBoost	Keras Deep Learning	Winner?
Performance on test set			
Accuracy	80.1%	78.1%	XGBoost
recall: true positive / (true positive + false negative)	0.89	0.68	
false negatives	1,200	3,500	
Training time	1 minute 24 seconds	2 minutes – 3 minutes for experiment 5 depending on hw env and patience setting	Inconclusive – deep learning training time varies
Code complexity	• Extra steps required to transform data coming out of pipeline • 1 line to build model	• Data from pipeline ready to train model • Complex model build	Inconclusive
Flexibility	Handles continuous & categorical columns	Handles continuous, categorical, text and BLOB columns	Deep learning

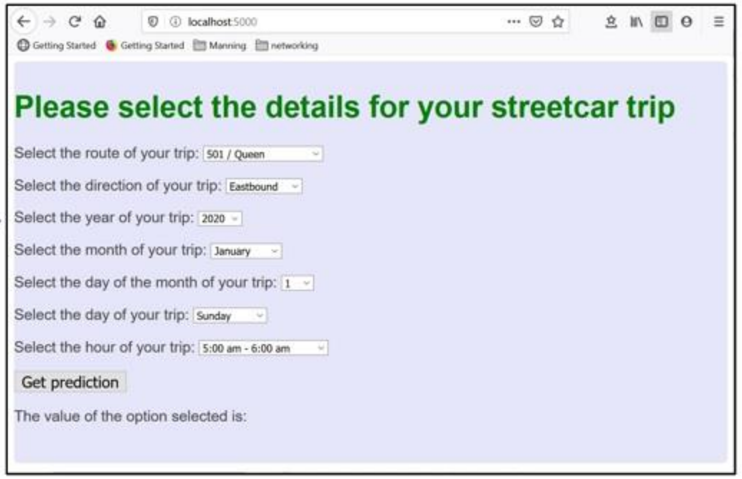
Web Deployment

- *Trained Keras model (h5 file)*
- *Custom pipeline classes (.py file)*
- *Pipeline (pkl file)*

Flask

- Web deployment using Python web server Flask

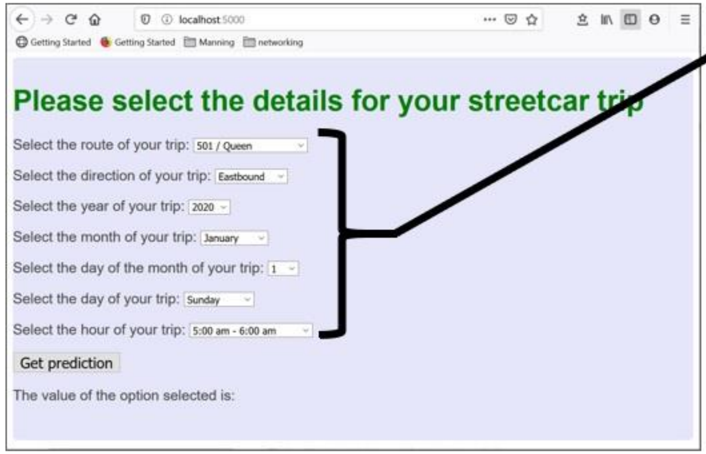
Web pages (HTML & CSS files)



Web Deployment: Step by Step

Score new data points with web deployment

home.html



show-prediction.html

1. User selects trip details (scoring parameters) in `home.html` and clicks on **Get prediction**

2. Javascript functions `getOption()` and `link_with_args()` in `home.html`:
   - Format the scoring parameters as a an argument string that gets added to the URL for `show_prediction`
   - Fire link to show_prediction including scoring parameters, e.g.

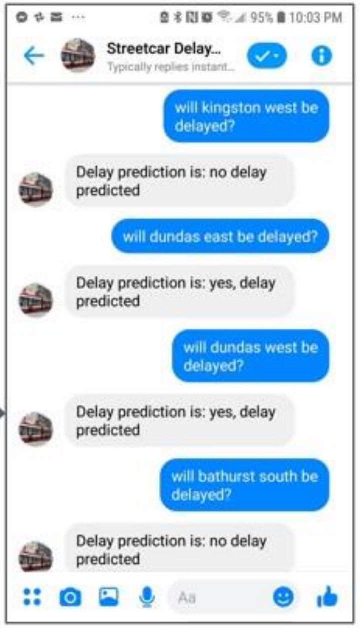
   `/show-prediction/?route=501&direction=e&year=2019&month=1&daym=1&day=6&hour=5`

3. Code in `flask_server.py` catches the link to `show-prediction` and:
   - Parses the scoring parameters from the URL into a dataframe
   - Loads the pipelines and the trained model
   - Applies the pipelines and the trained model to the dataframe containing the scoring parameters to get a prediction
   - Launches `show-prediction.html` with the prediction value as an argument

4. `show-prediction.html` is displayed with the prediction for the trip the user entered in `home.html`

Facebook Messenger Deployment

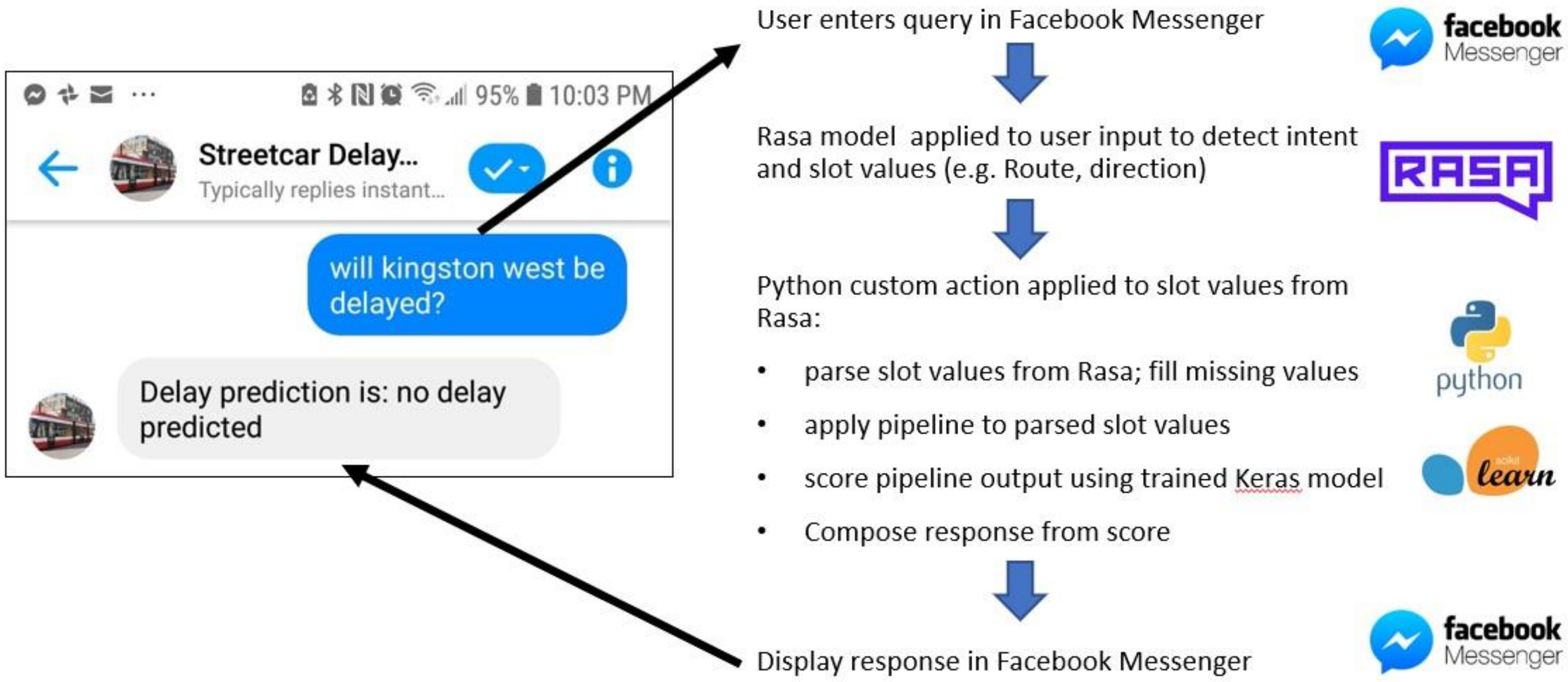
- Facebook Messenger deployment with Rasa chatbot



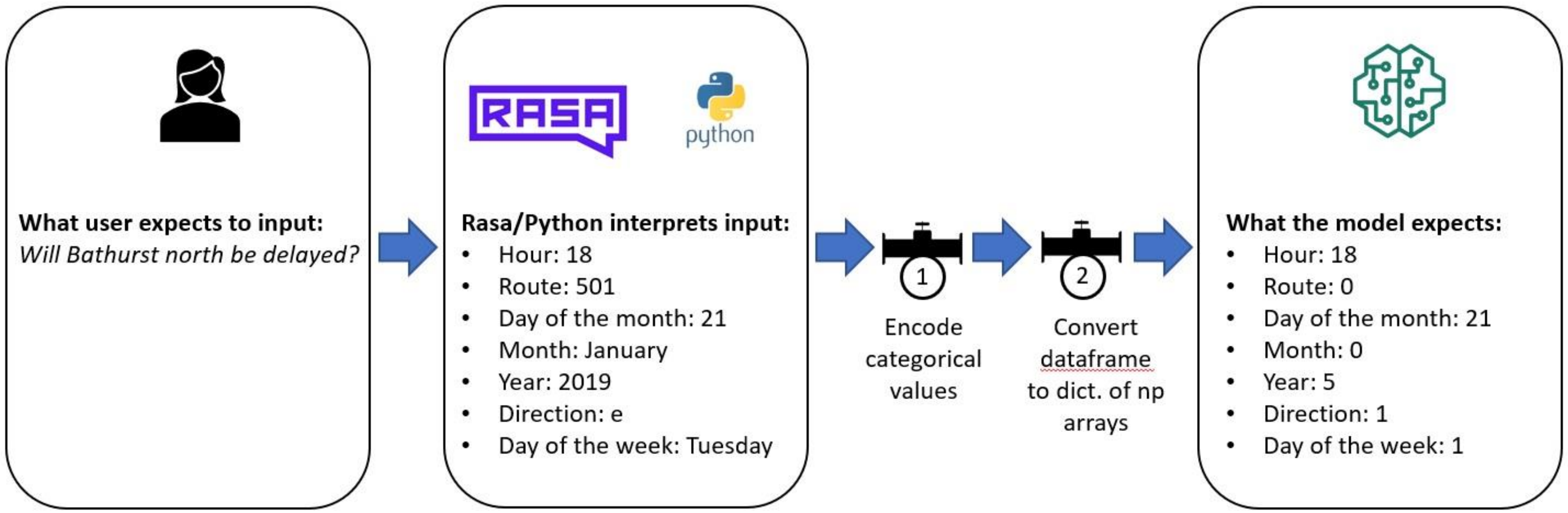
- *Trained Keras model (h5 file)*
- *Custom pipeline classes (.py file)*
- *Pipeline (pkl file)*

- *Rasa model & actions (.py file)*
- *Facebook config settings*

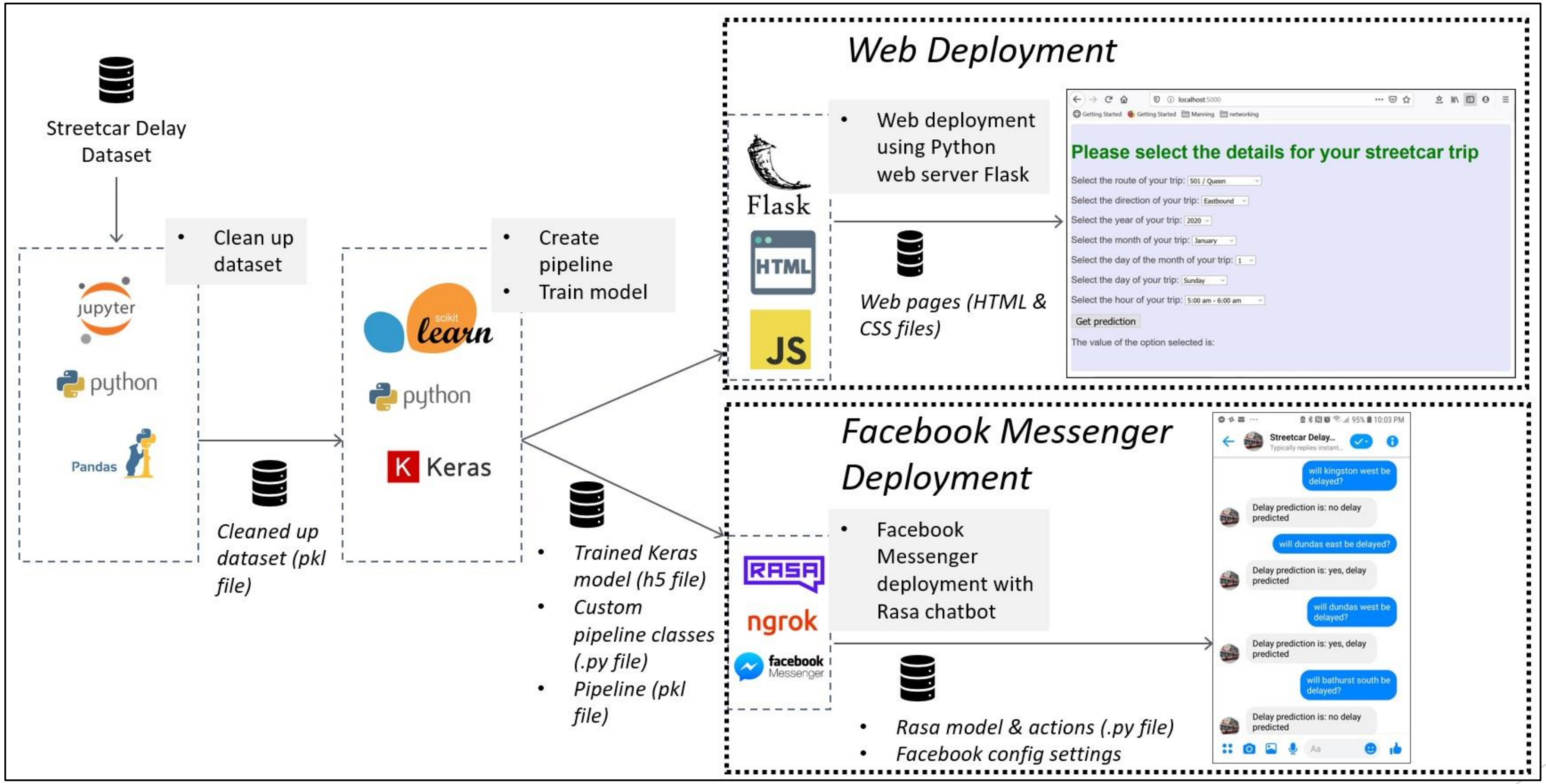
Facebook Messenger Deployment: Step by Step



Pipeline from Training Used in Deployment

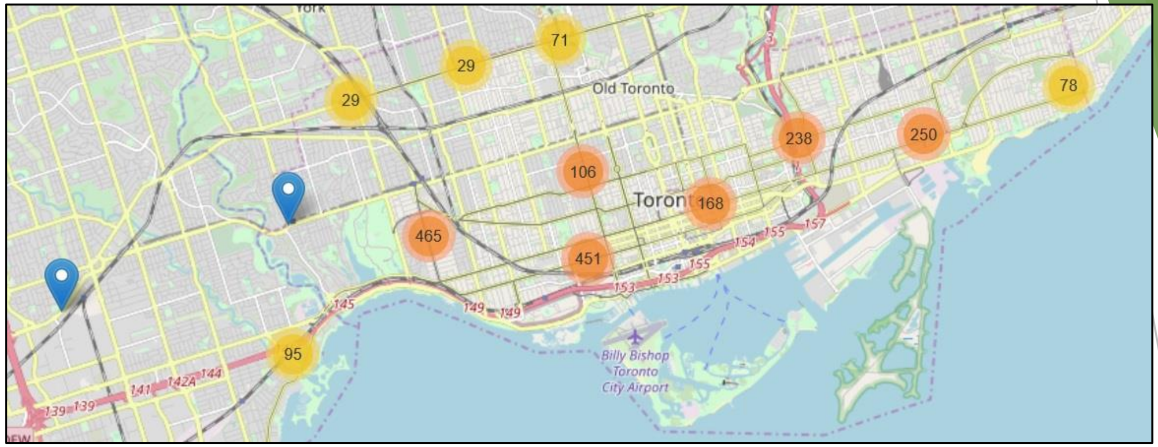


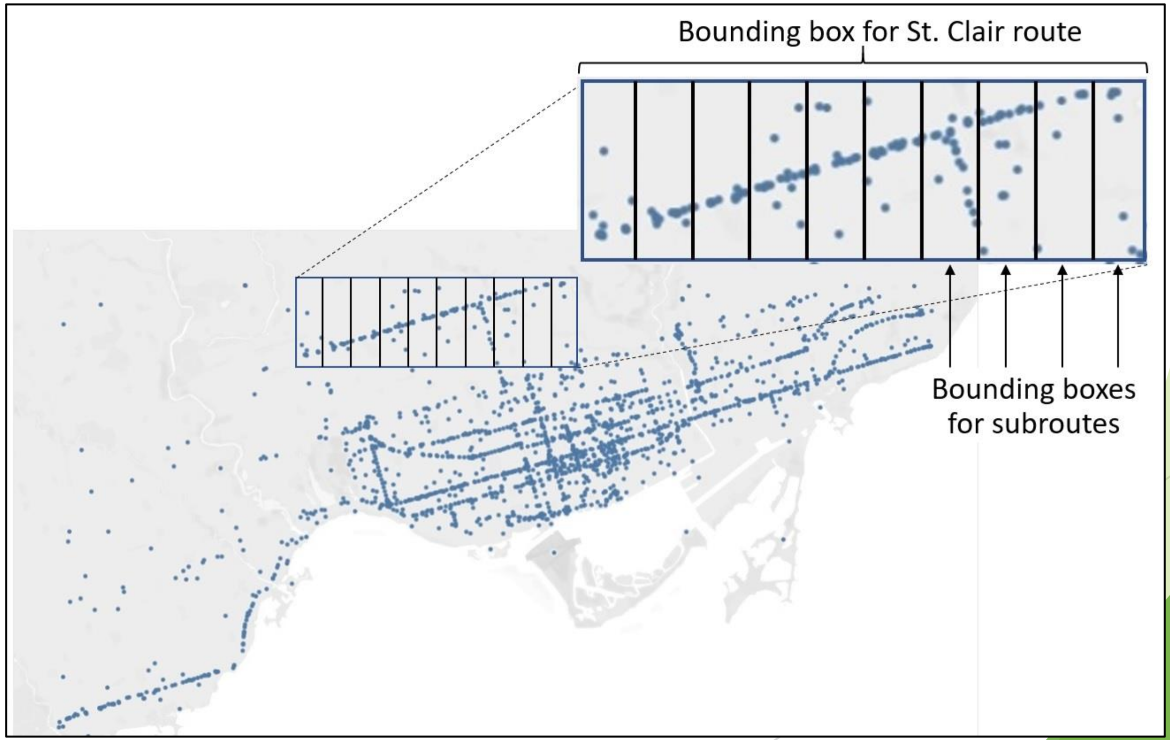
Simple but end-to-end



Next steps

- Add geospatial data
- Add weather data
- Re-implement in fastai
- Apply the same approach to the other datasets (e.g. Airbnb NYC)



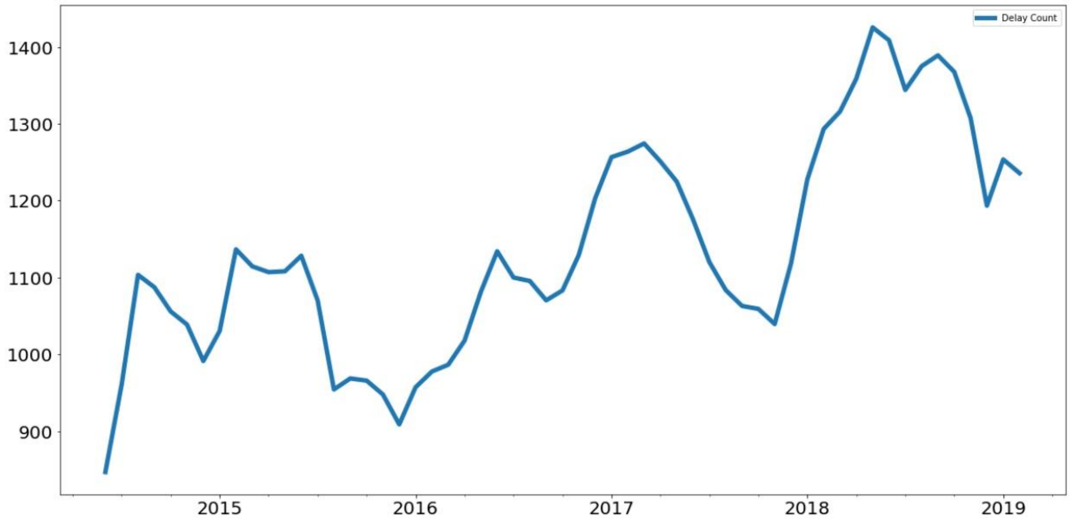


Resources:

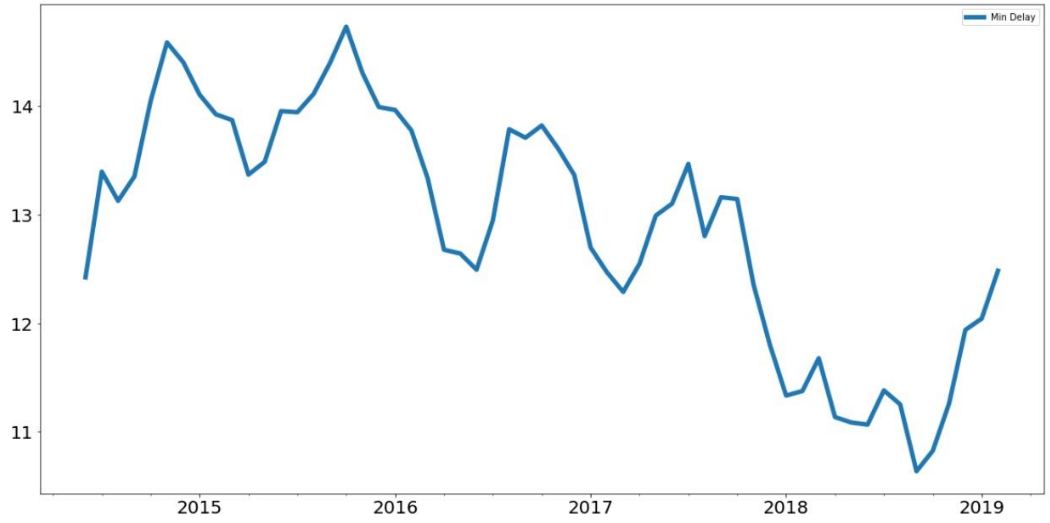
- Repo accompanying the book: https://github.com/ryanmark1867/deep_learning_for_structured_data
- Book site: https://www.manning.com/books/deep-learning-with-structured-data
- RAPIDS: https://developer.nvidia.com/rapids
- fast.ai course: https://course.fast.ai/
- TabNet
- Some examples of research on deep learning with structured data: https://scholar.sun.ac.za/handle/10019.1/106113; https://arxiv.org/abs/1805.06440
- Connect with me:
  - LinkedIn: https://www.linkedin.com/in/mark-ryan-31826743/
  - Medium: https://medium.com/@markryan_69718

BACKUP

Explore the Data



Delay count trend

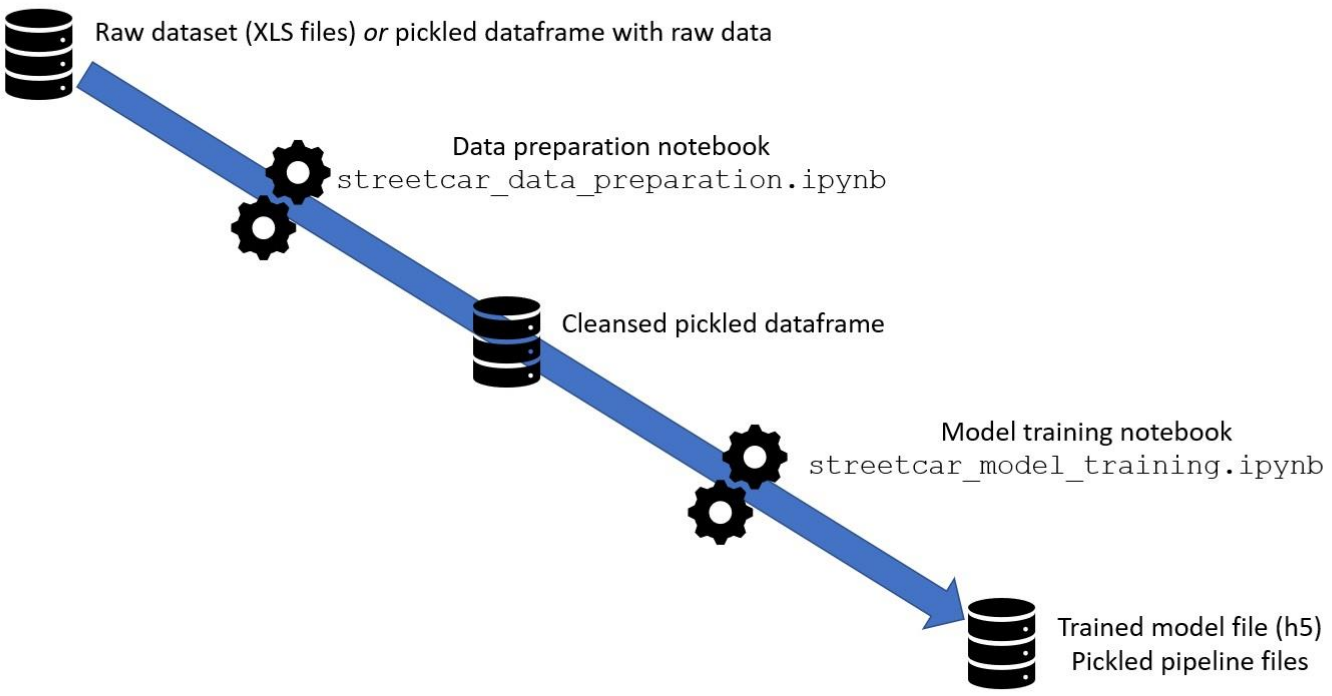


Delay duration trend

Repo Code Structure

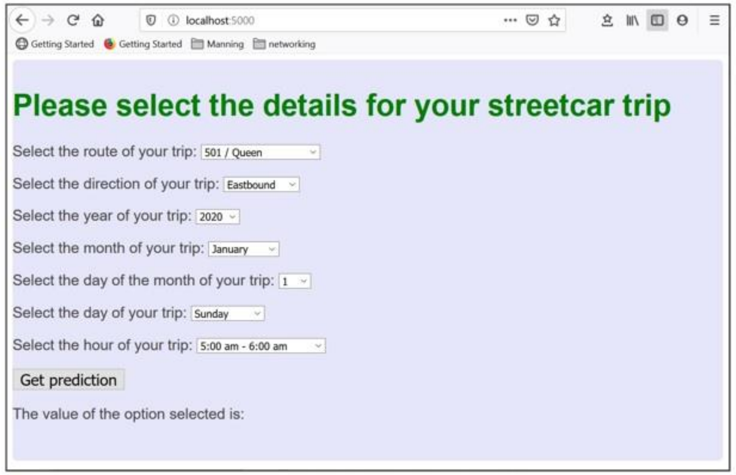
```
├── data              →  Datasets and pickled dataframes
├── deploy            →  Facebook Messenger deployment
│   ├── data
│   ├── input_sample_data
│   └── models
├── deploy_web        →  Web deployment
│   ├── static
│   │   └── css
│   └── templates
├── models            →  Trained Keras and XGBoost models
├── notebooks         →  Code for data cleanup & model training
├── pipelines         →  Trained pipelines
└── sql               →  SQL to create a sample table
```

Code Flow 1: Raw Data to Trained Model



Code Flow 2: Trained Model to Deployment

Trained model file (h5)
Pickled pipeline files



Web deployment
`flask_server.py`
`home.html`
`show-prediction.html`

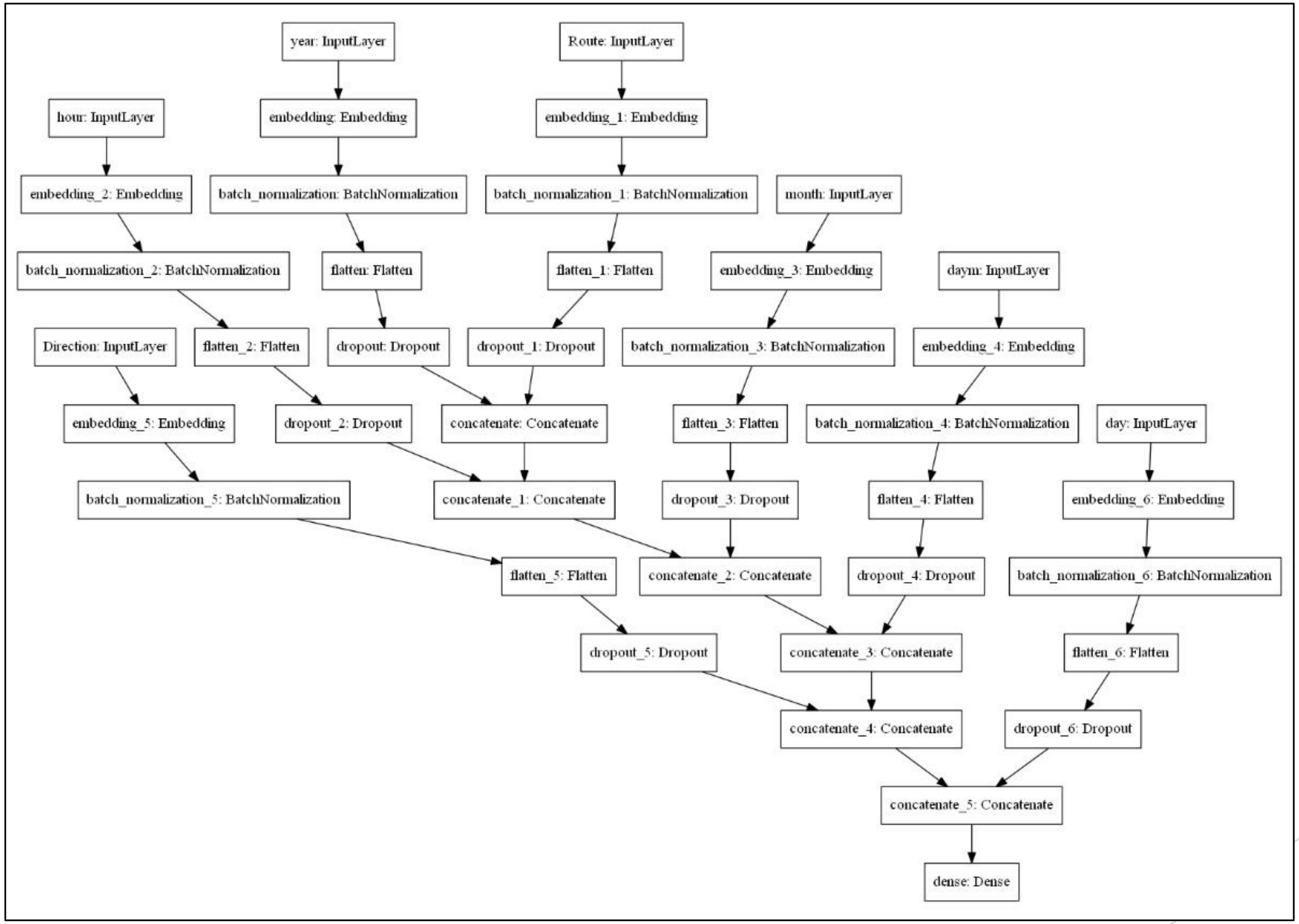
Facebook Messenger deployment
Rasa chatbot training & config files
`actions.py`

Useful Coding ideas

- Config files
- Logging
- Pickle files to serialize intermediate datasets

```
general:
    load_from_scratch: False
    save_transformed_dataframe: True
    remove_bad_values: True
file_names:
    pickled_input_dataframe: 2014_2019.pkl
    pickled_output_dataframe:
    2014_2019_df_cleaned_remove_bad_values_apr5_2020.pkl
```

Build Model: Keras Model Layers



Train the Model using Keras Callbacks