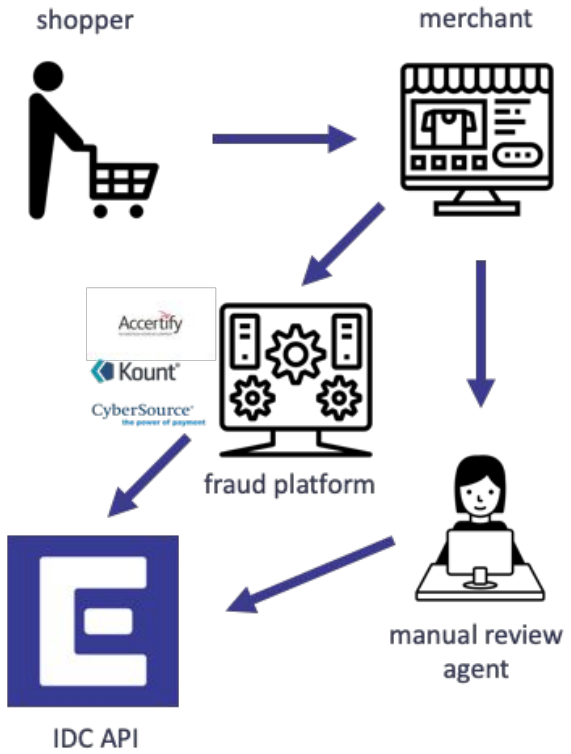
Model packaging

Gyorgy Mora - Ekata

long version here

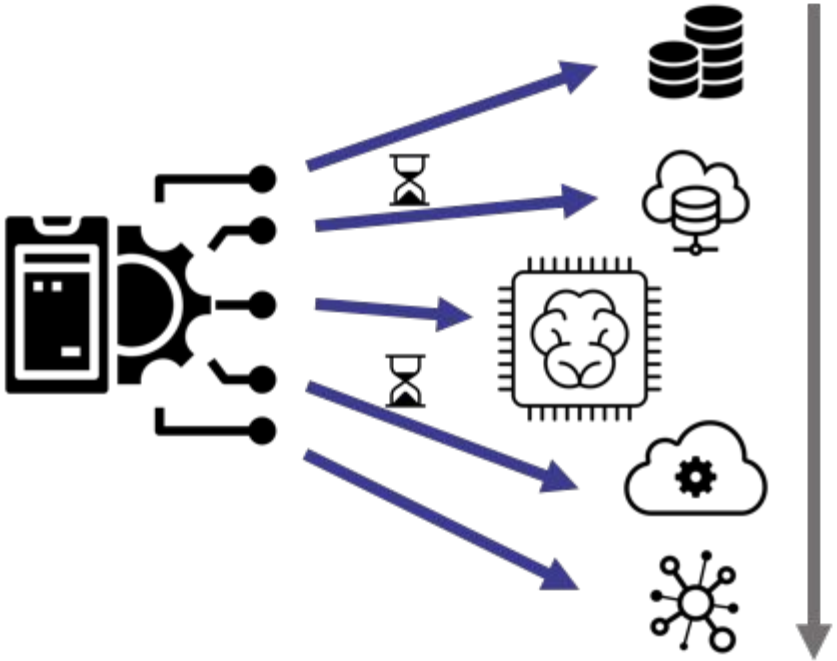
Identity verification - fraud prevention

- Identity
  - E-mail
  - Phone
  - Address
  - IP address
- Identity Graph database
- Transaction history database



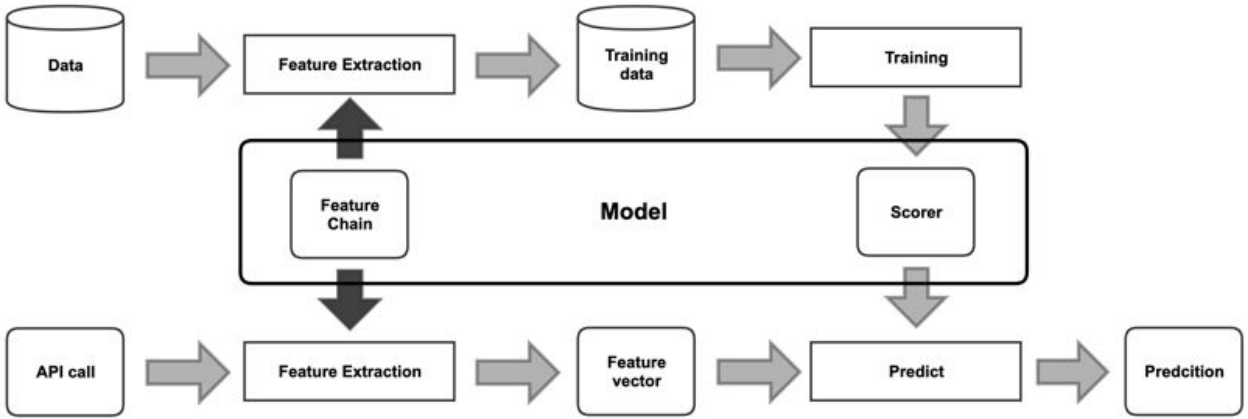
Why packaging?

- Prediction during checkout
- Existing 500 ms API product
  - Score got 10ms
- JVM backend
- System design from existing API
- Limited engineering resource



One json to rule them all

- Feature descriptions
  - Nominal mapping
  - Frequencies
  - Basic transformations
- Tree descriptions



```
{
 "features:": [
   {
      "type": "frequency",
      "name": "carrier",
      "frequncymap": "us_carrier",
     "buckets": [ 0.1, 0.01, 0.001 ]
   }
 ],
 "frequencies": {
   "us_carriers": {
     "AT&T": 0.124,
     "Vodafone": 0.119,
     "T-Mobile": 0.193
     "Verizon": 0.0131,
     "US Cellular": 0.00102
   }
 }
}
```

```
{
 "type": "xgboostedtree",
 "positiveclass": 1,
 "attributenum": 71,
 "classnum": 2,
 "minfeaturenum": 3,
 "trees": [
   {
     "split": 25,
     "value": 0.999,
     "yes": {
     "posteriori": [
          "NaN",
          "-0.2345"
     ],
     },
     "no": {...},
     "missing": {...},
     "numeric": true
   },
...
```

```
override fun getDoubleValue(featureValue: String?): Double {
  return if (featureValue != null) {
    val fvRaw = if (lowerCase) {
      featureValue.toLowerCase()
    } else {
      featureValue
    }

    val fv = valueReverseMap[fvRaw]

    if (fv != null && attributeValues.contains(fv)) {
      attribute.valueOf(fv)
    } else if (otherValue != null) {
      attribute.valueOf(otherValue)
    } else {
      log.warn("Unknown value for feature ${attribute.name} : $fv ($fvRaw)")
      Double.NaN
    }
  } else {
    Double.NaN
  }
}
```

Getting the model out

```
booster.getModelDump("", true, "json")
```

- Pseudo Json
- One line per tree
- No feature names just numbers
- Statistics
- <=

```
[
 { "nodeid": 0, "depth": 0, "split": 28, "split_condition": -9.53e-07, "yes": 1, "no": 2, "missing": 1, "gain": 4000.53101,
"cover": 1628.25, "children": [
   { "nodeid": 1, "depth": 1, "split": 55, "split_condition": -9.53674316e-07, "yes": 3, "no": 4, "missing": 3, "gain":
1158.21204, "cover": 924.5, "children": [
     { "nodeid": 3, "leaf": 1.71217716, "cover": 812 },
     { "nodeid": 4, "leaf": -1.70044053, "cover": 112.5 }
   ]},
   { "nodeid": 2, "depth": 1, "split": 108, "split_condition": -9.53674316e-07, "yes": 5, "no": 6, "missing": 5, "gain":
198.173828, "cover": 703.75, "children": [
     { "nodeid": 5, "leaf": -1.94070864, "cover": 690.5 },
     { "nodeid": 6, "leaf": 1.85964918, "cover": 13.25 }
   ]}
 ]},
 { "nodeid": 0, "depth": 0, "split": 59, "split_condition": -9.53674316e-07, "yes": 1, "no": 2, "missing": 1, "gain":
832.545044, "cover": 788.852051, "children": [
   { "nodeid": 1, "depth": 1, "split": 28, "split_condition": -9.53674316e-07, "yes": 3, "no": 4, "missing": 3, "gain":
569.725098, "cover": 768.389709, "children": [
     { "nodeid": 3, "leaf": 0.78471756, "cover": 458.936859 },
     { "nodeid": 4, "leaf": -0.968530357, "cover": 309.45282 }
   ]},
   { "nodeid": 2, "leaf": -6.23624468, "cover": 20.462389 }
 ]}
]
```

Predictor features

- Explaining predictions (reason codes)
- Handle missing values
- Smoothing decision surfaces
- Target language native code
- Parallelized

We do open source

- Basic feature primitives
- Training
  - Spark/JVM
  - Python
- Prediction
  - JVM
  - Python