Diabetes Case Study

March 11, 2020

1 Diabetes Case Study

In this case study, we take the Pima Indians Diabetes Dataset from Kaggle/UCI Machine Learning to Predict the onset of diabetes based on diagnostic measures and showcase the Monitaur platform.

1.0.1 Dataset:

https://www.kaggle.com/uciml/pima-indians-diabetes-database

1.1 About Monitaur

Monitaur (https://www.monitaur.ai/) is a machine learning assurance platform that enables recording, understanding, verification, and auditing of your machine learning models. The platform is environment agnostic, with support for all classical machine and deep learning implementations in Python.

For companies in regulated industries using models to make decisions, Monitaur delivers transparency and auditability that's necessary to manage compliance and unlock innovation.

1.1.1 Product details

What does Monitaur record?

- Model inputs
- Model prediction
- Model version
- Meta information, such as developer name, owner name, library used, model used

What can we do with the captured information?

- Allow verifiability of machine learning predictions and create an audit trail
- Check when a model has changed
- Provide transaction interpretability
- Enable audibility of models with the ability to rerun transactions
- Detect when model and feature drift occur
- Provide automated model documentation and reporting for a machine learning model

1.1.2 Core Components

Monitaur has four core components:

1. Record
2. Understand
3. Verify
4. Audit

Record The Monitaur client library is designed to record your machine learning model's meta information along with hashed versions of the serialized model and production files. Each transaction is then recorded as it runs on your production infrastructure.

In addition, Monitaur versions your models based on changes to the production and trained model file hashes. When changes occur, the platform automatically generates alerts. You can also define your own alerting logic based on transaction-level details.

Understand As transactions are sent to the back-end API, Monitaur obtains the underlying influences for each of your model's decisions via open source interpretability libraries. This, along with the ability to create configurable alerts, Monitaur provides insight into all of your model needs.

Verify The Monitaur GRC (Governance, Risk, and Compliance) web application is designed specifically for the non-technical user to find transactions.

Audit With full model versioning and transaction reproducibility, the platform allows counterfactual exploration for auditibility of machine learning models. With metrics for model and feature drift, along with proactive bias controls, Monitaur can provide automated model audit reports. Inside Monitaur, we provide model whitepaper creation and workflow processes around proven machine learning governance frameworks.

[1]:
```
# Install libraries

import pandas as pd
import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from xgboost.sklearn import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from joblib import dump, load
import numpy as np
```

[2]:
```
# Check sklearn version
sk.__version__
```

[2]:
```
'0.21.3'
```

1.2 Load data and review data

[3]:
```
# Load data
dataset = pd.read_csv('data/diabetes.csv')
```

[4]:
```
# Review data
dataset.head()
```

[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[5]:
```
featureNames = dataset.columns.tolist()
```

[6]:
```
# remove outcome
featureNames.pop()
```

[6]:
```
'Outcome'
```

[7]:
```
featureNames
```

[7]:
```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age']
```

[8]:
```
# Split data into X and y
X = dataset.drop('Outcome',axis=1).values
Y = dataset['Outcome'].values
```

[13]:
```
# Split data into train and validation sets
seed = 10
test_size = 0.05
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=test_size,␣
 ↪random_state=seed)
```

[14]:
```
# Create pipeline
xgboost_pipeline = Pipeline([('scaler', StandardScaler()), ('xgboost',␣
 ↪XGBClassifier())])

# Note: Scaling (removing the mean and scaling to unit variance of features)␣
 ↪isn't necessary for XGBoost,
# but we are including it anyway to illustrate a pipeline.
```

[15]:
```
# Tune the model with a cross validation grid search
parameters = {
    "xgboost__learning_rate": [0.01, 0.05, 0.10,0.2,0.3],
     "xgboost__max_depth": [ 3, 4, 5, 6, 8, 10, 12, 15],
     "xgboost__gamma": [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
}


gridsearch = GridSearchCV(xgboost_pipeline,
                          parameters,
                          cv=5,
                          n_jobs=-1,iid=False)
# Fit pipeline
gridsearch.fit(X_train, y_train)
print("Best parameter (CV score=%0.3f):" % gridsearch.best_score_)
print(gridsearch.best_params_)
```

```
Best parameter (CV score=0.769):
{'xgboost__gamma': 0.2, 'xgboost__learning_rate': 0.05, 'xgboost__max_depth': 3}
```

[16]:
```
# Make predictions for test data
y_pred = gridsearch.predict(X_val)
```

[17]:
```
# Evaluate predictions
target_names = ['You have diabetes', 'You do not have diabetes']

print(classification_report(y_val, y_pred, target_names=target_names))
```

```
                          precision    recall  f1-score   support

      You have diabetes       0.75      0.81      0.78        26
You do not have diabetes       0.55      0.46      0.50        13
```

accuracy			0.69	39
macro avg	0.65	0.63	0.64	39
weighted avg	0.68	0.69	0.69	39

[18]:
```
# Save the model
dump(gridsearch, 'DiabetesPipeline.joblib')
```

[18]:
```
['DiabetesPipeline.joblib']
```

1.3 Integrate Model with Monitaur

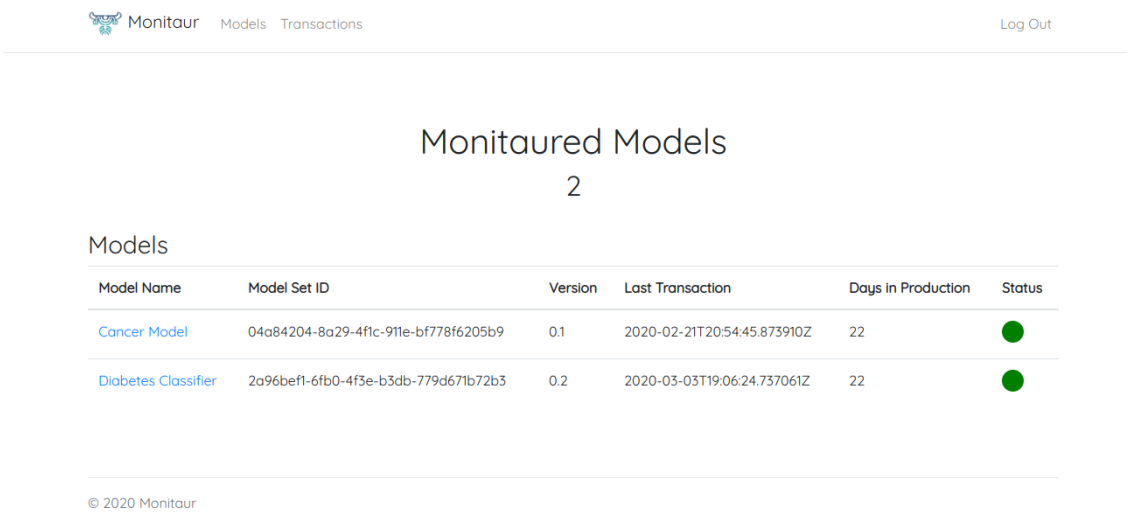
[19]:
```
# Import Monitaur library
from monitaur import Monitaur
from monitaur.utils import hash_file
import platform

# Monitaur authentication token - When you on-board with Monitaur, we provide␣
 ↪an authentication token
# and access to our web user interface. We are deployable on-prem, in your␣
 ↪cloud, or through our SaaS platform.
monitaur = Monitaur(
    auth_key="6f6456b1758280339ff141bd0cc902c06cca8585",
    base_url="https://monitaur-api-dev.herokuapp.com",
)
```

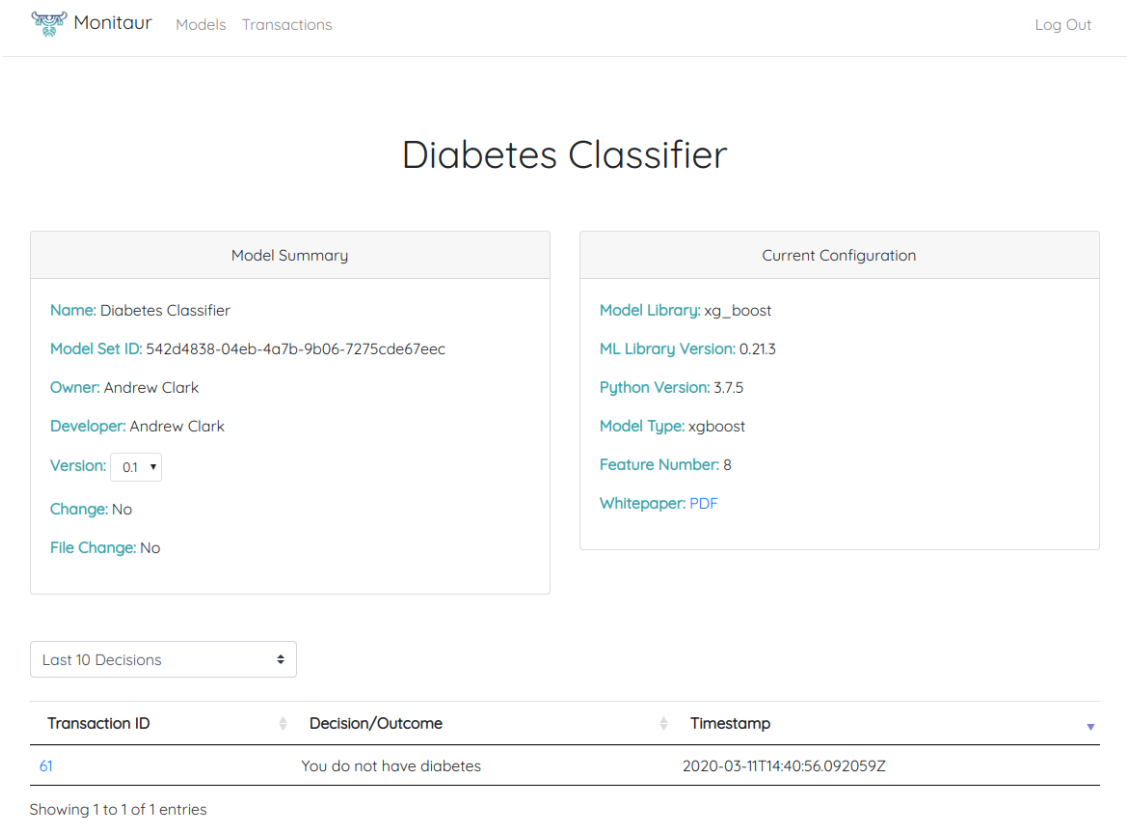
[20]:
```
# Add model to Monitaur and we will return its unique idenifier, called a␣
 ↪model_set_id.
model_data = {
    "name": "Diabetes Classifier",
    "model_type": "xgboost",
    "model_class": "tabular",
    "library": "xg_boost",
    "trained_model_hash": None,
    "production_file_hash": None,
    "feature_number": 8,
    "owner": "Andrew Clark",
    "developer": "Andrew Clark",
    "python_version": platform.python_version(),
    "ml_library_version": sk.__version__,
    "influences": True,
}
model_set_id = monitaur.add_model(**model_data)
```

```
201
{'_content': b'{"id":45,"model_type":"xgboost","library":"xg_boost","last_transa
```

ction":"N/A","python_version":"3.7.5","ml_library_version":"0.21.3","days_in_prod":0,"status":"green","model_set_id":"542d4838-04eb-4a7b-9b06-7275cde67eec","name":"Diabetes Classifier","model_class":"tabular","trained_model_hash":null,"production_file_hash":null,"feature_number":8,"owner":"Andrew Clark","developer":"Andrew Clark","current":true,"version":0.1,"influences":true,"whitepaper":"https://drive.google.com/file/d/1Hd_DZiyiPcZ6A1h1AT2Z8diur66KyjTn/view?usp=sharing","created_date":"2020-03-11T14:40:35.895546Z","updated_date":"2020-03-11T14:40:35.895586Z"}', '_content_consumed': True, '_next': None, 'status_code': 201, 'headers': {'Connection': 'keep-alive', 'Server': 'gunicorn/20.0.4', 'Date': 'Wed, 11 Mar 2020 14:40:35 GMT', 'Content-Type': 'application/json', 'Vary': 'Accept', 'Allow': 'GET, POST, HEAD, OPTIONS', 'X-Frame-Options': 'DENY', 'Content-Length': '632', 'Strict-Transport-Security': 'max-age=3600; includeSubDomains; preload', 'X-Content-Type-Options': 'nosniff', 'X-Xss-Protection': '1; mode=block', 'Referrer-Policy': 'same-origin', 'Via': '1.1 vegur'}, 'raw': <urllib3.response.HTTPResponse object at 0x7f0bcfe5dc10>, 'url': 'https://monitaur-api-dev.herokuapp.com/api/models/', 'encoding': None, 'history': [], 'reason': 'Created', 'cookies': <RequestsCookieJar[]>, 'elapsed': datetime.timedelta(microseconds=320118), 'request': <PreparedRequest [POST]>, 'connection': <requests.adapters.HTTPAdapter object at 0x7f0bcfeae410>}

Below you can see the model added on the home and model detail pages of Monitaur's Goverance, Risk, and Controls interface





[21]:
```
# Get aws credentials
credentials = monitaur.get_credentials(model_set_id)
```

[22]:
```
# Record training
# If you want to have Anchors understanding for your tabular model, run the␣
 ↪record_training API call.
record_training_data = {
    "credentials": credentials,
    "model_set_id": model_set_id,
    "trained_model": gridsearch,
    "training_data": X_train,
    "feature_names": featureNames,
    "re_train": False
}
monitaur.record_training(**record_training_data)
```

```
Training recording: model_set_id 542d4838-04eb-4a7b-9b06-7275cde67eec, version
0.1
```

[22]:
```
True
```

1.4 Prediction

We are showing a production, i.e. individual prediction, deployment in this notebook, although this code would normally be in a seperate, normally .py file.

[23]:
```
model = load('DiabetesPipeline.joblib')
```

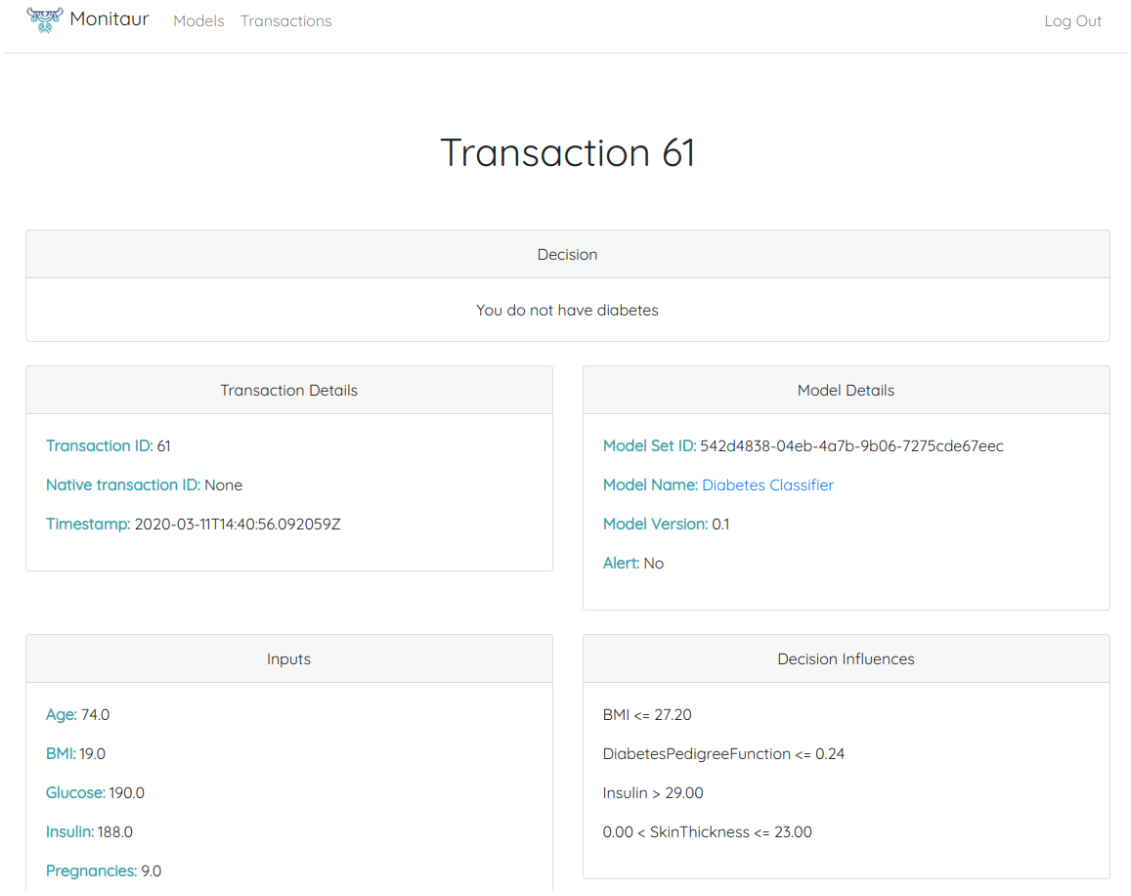
[24]:
```
# Create random data inputs for example
Pregnancies = np.random.randint(0,10, size=1)[0]
Glucose = np.random.randint(160,200, size=1)[0]
BloodPressure = np.random.randint(70,100, size=1)[0]
SkinThickness = np.random.randint(10,30, size=1)[0]
Insulin = np.random.randint(180,200, size=1)[0]
BMI = np.random.randint(19,40, size=1)[0]
DiabetesPedigreeF = round(np.random.uniform(0,1,size=1)[0],2)
Age = np.random.randint(1,100,size=1)[0]


# Create a numpy array off of the inputs
data = np.array([[Pregnancies, Glucose, BloodPressure, SkinThickness,
                      Insulin, BMI, DiabetesPedigreeF, Age]])

# Predict if the individual has diabetes or not.
result = model.predict(data)

if result[0] == 1:
    prediction = 'You have diabetes'
else:
    prediction = 'You do not have diabetes'

prediction
```

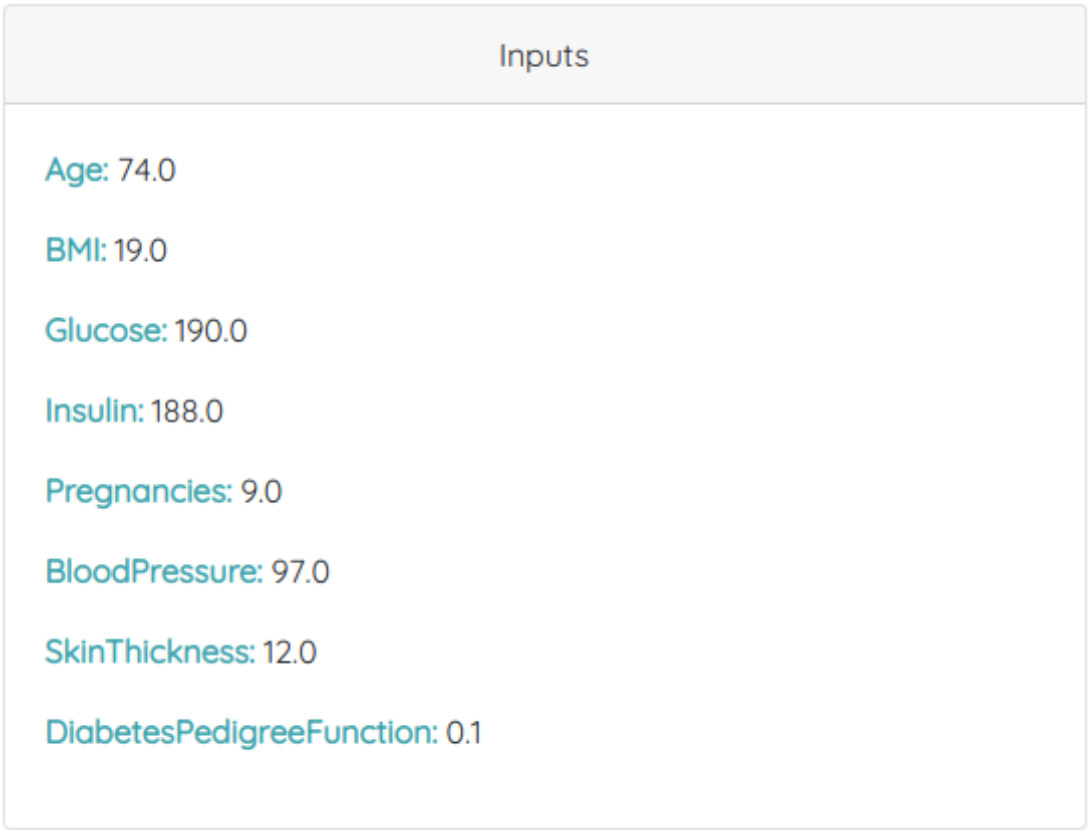
[24]:
```
'You do not have diabetes'
```

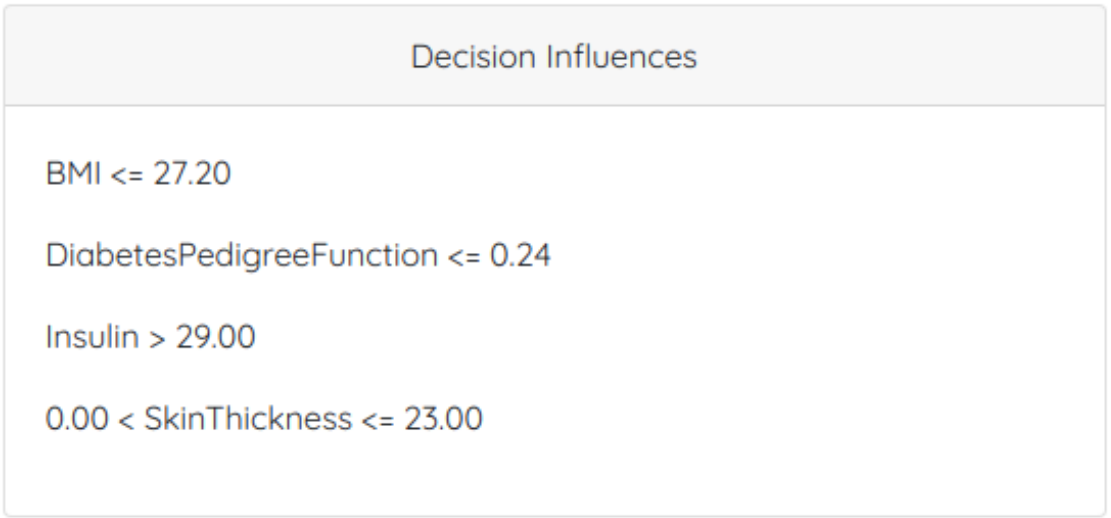
[25]:
```
# Take the list of feature names and data inputs and zip them into a python␣
 ↪dictionary for the API call.
dictionaryFeatures = dict(zip(featureNames,data.tolist()[0]))

transaction_data = {
    "credentials": credentials,
    "model_set_id": model_set_id,
    "trained_model_hash": hash_file('DiabetesPipeline.joblib'),
    "production_file_hash": hash_file("Diabetes Case Study.ipynb"),
    "prediction": prediction,
    "features": dictionaryFeatures,
}
response = monitaur.record_transaction(**transaction_data)
```

```
Could not find an anchor satisfying the 0.95 precision constraint. Now returning
the best non-eligible anchor.
```

Below we can see on the Monitaur Goverance, Risk, and Controls interface showing that the transaction was recorded. We can see the inputs as well as the Anchor's local interpretability of the transaction.







1.5 View transactions

Monitaur provides the ability to programmatically access platforms through the read_transactions API call.

[26]:
```
transactions = monitaur.read_transactions(model_set_id=model_set_id)
```

[27]:
```
# bring the transactions into a pandas dataframe for ease of use.
df = pd.DataFrame(transactions)
df
```

[27]:

```
   id                           model_set_id  \
0  61  542d4838-04eb-4a7b-9b06-7275cde67eec

                                              features  \
0  {'Age': 74.0, 'BMI': 19.0, 'Glucose': 190.0, '…

                                     interpretability  \
0  ["BMI <= 27.20", "DiabetesPedigreeFunction <= …

                                   trained_model_hash  \
0  cb87893e8873320cf35e83e9026f10de134a3f9f53f369…

                                  production_file_hash  \
0  9d16d5827eb59e8f6657c2c2ccc8f7ab5c6d7c4da5d943…

                 prediction image native_transaction_id  \
0  You do not have diabetes  None                  None

                   created_date                 updated_date  model  \
0  2020-03-11T14:40:56.092059Z  2020-03-11T14:40:56.092105Z     45

            model_name  alerts
0  Diabetes Classifier       0
```

1.6 Conclusion

During this Case Study, we've created a Diabetes Classifier to be used for illustrative purposes if someone has diabetes or not. We also introduced Monitaur, and showed how easy it is to add a model and record transactions with our API. If you would like to learn more about Monitaur and how we can help provide assurance around your machine learning implementations to help unlock innovation, visit our website: Monitaur.ai and request a demo.

Andrew Clark, Co-founder and CTO of Monitaur.ai