Thielmann Analytics

CLAIMS FRAUD MODEL IMPLEMENTATION

OPERATIONAL REVIEW – FINAL PRODUCT

DECEMBER 2020

CONTENTS

EXHIBIT INTRO - EXECUTIVE SUMMARY

Regis University practicum course requested a project which applied techniques introduced throughout the degree program. Today's conversation will review the candidate product produced.

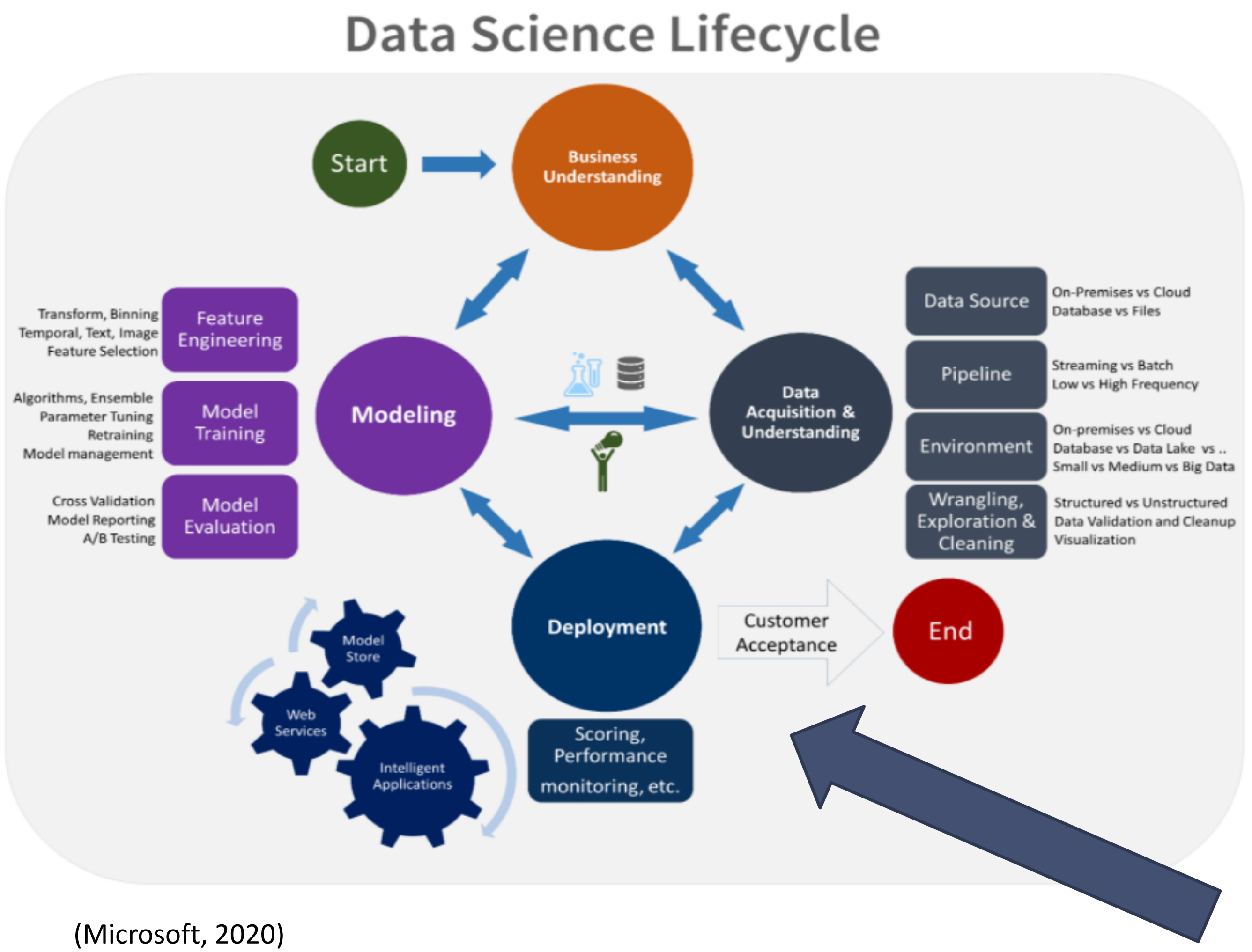
Objectives:

- Provide an overview of the implementation processes and methodologies
- Review the Claims Fraud Model final model project and new implementation techniques
- Discuss the business / team impact
- Review applicable next steps

Highlights:

- Improved model implementation method using API
- Automated notification process for independent feature drift
- Realtime dashboard for feature analysis
- Background process for distribution testing

EXHIBIT INTRO – Project Overview



(Microsoft, 2020)

Deployment and Implementation

- Model serialization
- Scoring function
- Realtime data preprocessing
- Model performance evaluation
- Realtime Dashboard
- Background Scheduling
- Distribution Drift Testing
- Email Generation

CONTENTS

CLAIMS FRAUD MODEL RECAP – Insurance Terminology

(Patient Advocate Foundation, n.d.)

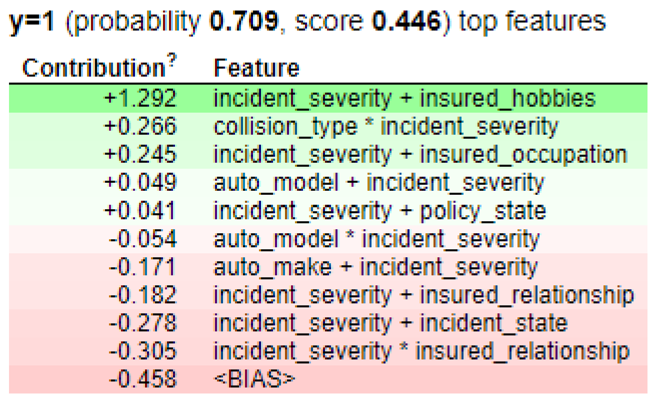
- **Policy:** Contract between insurer and policyholder. Determines claims insurer is legally required to pay
- **Premium:** Policyholder monthly payment
- **Deductible:** Amount spent by policyholder before insurer pays
- **Claim:** Formal request to insurer to receive compensation for a covered loss
- **Claims Adjuster:** Insurance employee responsible for investigating claims
- **Insurance Fraud:** Claimant attempts to obtain a benefit or advantage from the insurer to which they are not entitled

CLAIMS FRAUD MODEL RECAP - Overview

- 1000 rows of data
- Data split – 80% for Training, 20% for Testing
- Gradient Boosting method was used in final model
- Outlier removal with Gaussian Approximation
- Target Encoding for categorical features
- Deep Feature Synthesis using multiplicative and additive primitives
- Synthetic bootstrapping used to augment data samples
- Automated Pipeline for analysis assistance
- Model performance evaluated using lift and auc curve

CLAIMS FRAUD MODEL RECAP – Where we left off

Example of Fraudulent Claim



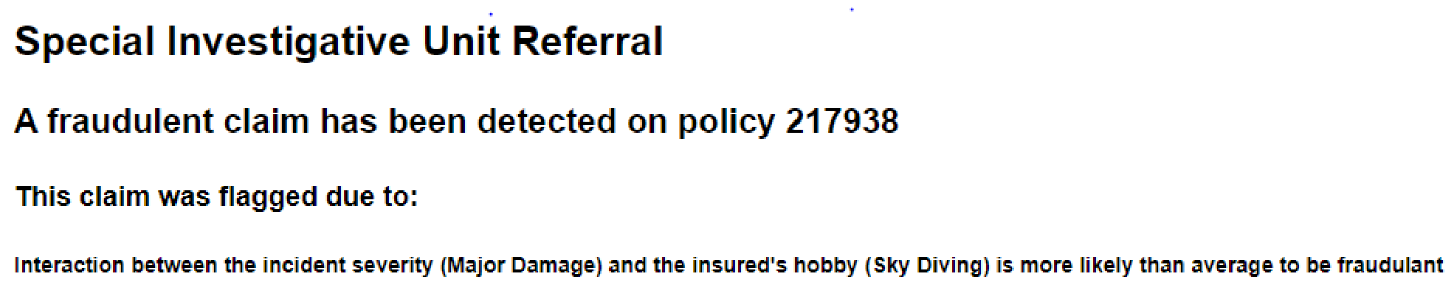
y=1 (probability **0.709**, score **0.446**) top features

Contribution?	Feature
+1.292	incident_severity + insured_hobbies
+0.266	collision_type * incident_severity
+0.245	incident_severity + insured_occupation
+0.049	auto_model + incident_severity
+0.041	incident_severity + policy_state
-0.054	auto_model * incident_severity
-0.171	auto_make + incident_severity
-0.182	incident_severity + insured_relationship
-0.278	incident_severity + incident_state
-0.305	incident_severity * insured_relationship
-0.458	<BIAS>

Based on the input, we can see that the top contributing feature in this prediction is the additive interaction between incident_severity and insured_hobbies

- **Model provides a prediction whether a claim is fraudulent or not as well as an explanation of possible reasons**
- **We now need to implement this model!**

Example of possible reason message:



Special Investigative Unit Referral

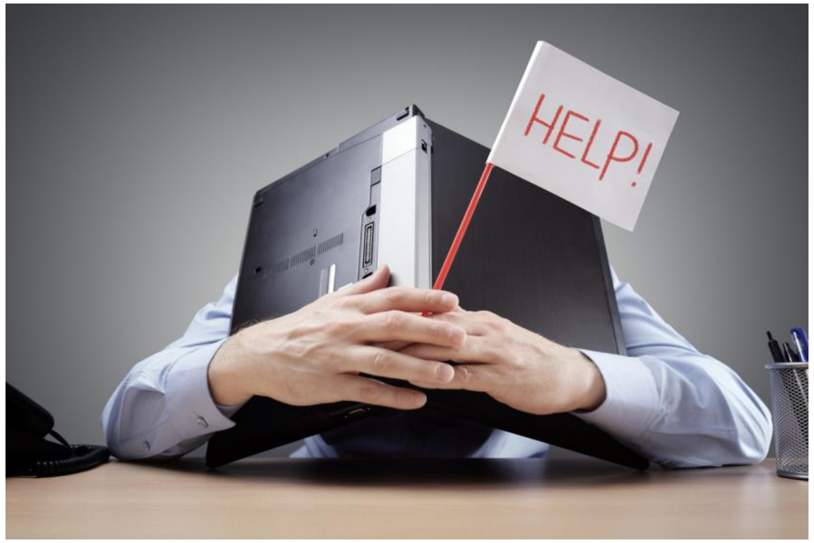
A fraudulent claim has been detected on policy 217938

This claim was flagged due to:

Interaction between the incident severity (Major Damage) and the insured's hobby (Sky Diving) is more likely than average to be fraudulant

CONTENTS

DEFINING AN API – Technical Warning!



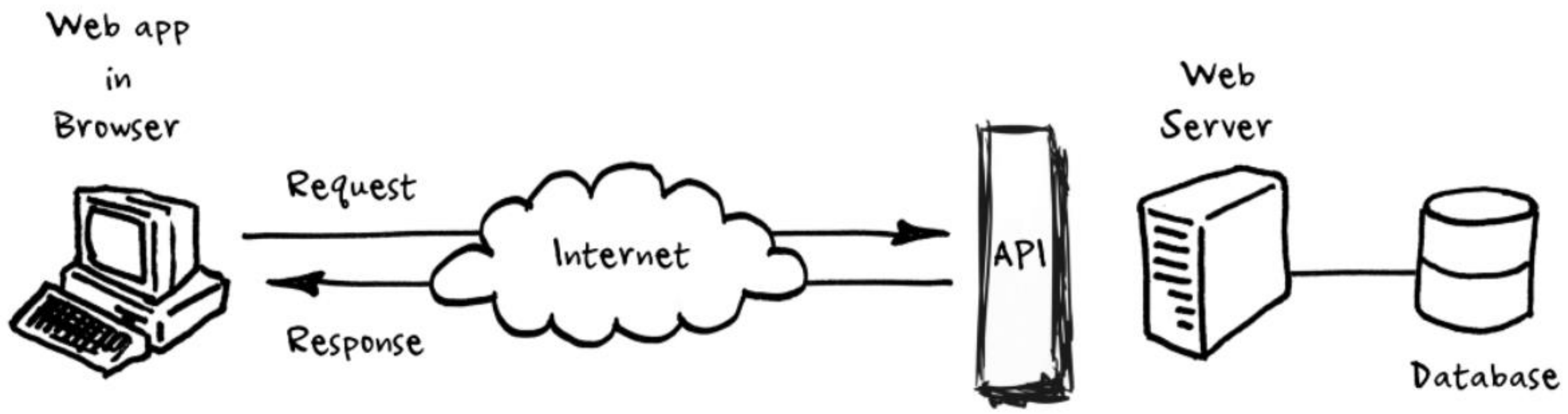
(Anderson, 2017)

- **Hang in there if things get confusing!**
  - I will attempt to elaborate on difficult topics
  - A section will be dedicated to WHY these techniques are important
- **Ask questions in the comments!**
  - With no in person attendance, please post any questions

(Discovery Place Nature, 2020)

Okay! Lets dive in…

DEFINING AN API– Application Programming Interface



(Eising, 2017)

What it is

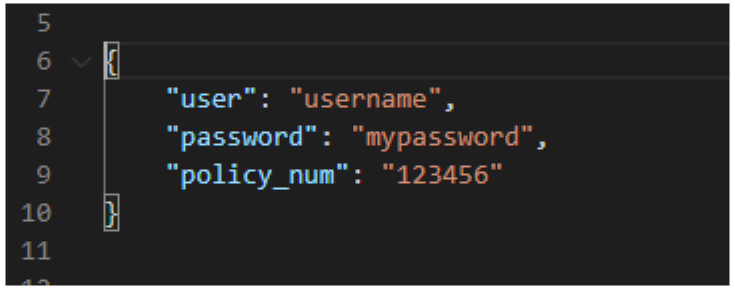
- Application Programming Interface (API)
- Allow applications to communicate with one another
- Provides the access point through which the applications pass data

The Benefit

- Automate the model scoring process
- Deliver a standardized communication method
- Client does not need to understand how model works

DEFINING AN API– Data Transfer Method: JSON

JSON Example



```
{
    "user": "username",
    "password": "mypassword",
    "policy_num": "123456"
}
```

What it is

- JavaScript Object Notation
- Data-interchange format
- Collection of Name: Value pairs

The Benefit

- Commonly used data exchange format
- Offers standardized method for app communication
- Human-readable for error troubleshooting

DEFINING AN API – Flask and Application Server

FLASK

- **Framework**
  - Provides the architecture for our application
- **Endpoints**
  - Provides a place the client application can interact with application features
  - Example:
    - @server.route("/score")
    - Becomes: http://127.0.0.1:5000/score

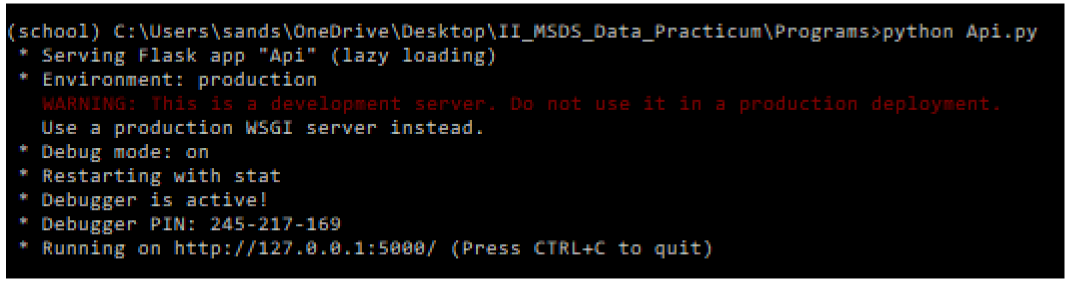
SERVER

- **Hardware**
  - Provides a place for our application to run

Flask
web development, one drop at a time

(Wikipedia, n.d.)

Application Running in Console



```
(school) C:\Users\sands\OneDrive\Desktop\II_MSDS_Data_Practicum\Programs>python Api.py
 * Serving Flask app "Api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 245-217-169
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

CONTENTS

APPLICATION FUNCTIONALITY – What does it all do?

Major Features

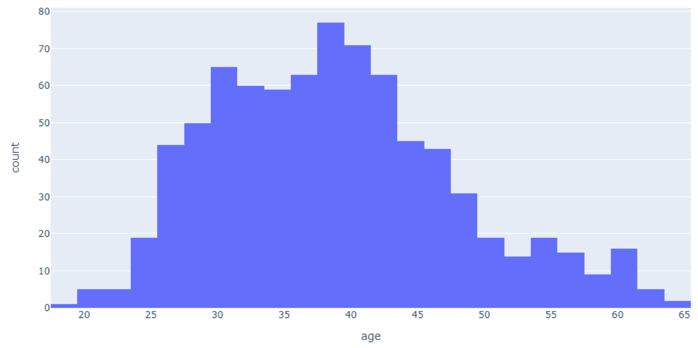
- **Background Process: Distribution Analysis**
  - Compares feature distributions every 24 hours
- **Score Function**
  - Ability to accept/return json for predictions
- **Dashboard**
  - Provide a real-time interactive visualization
- **Email**
  - Send email notification identifying significant data changes

Minor Features

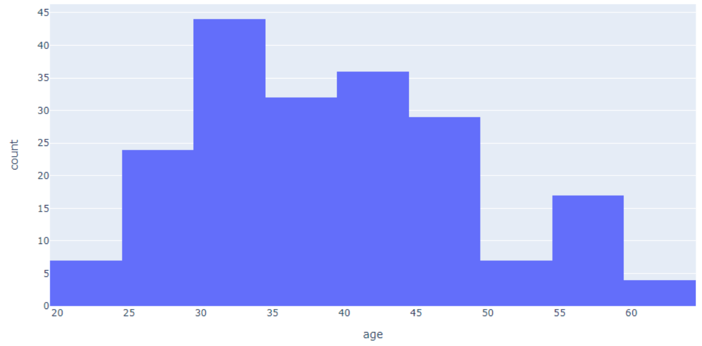
- **Database**
  - Created tables to stores application info such as prediction score runs
- **Minimal Data Input**
  - Application functions to pull data based on input ID data
- **Preprocess data**
  - Allows for raw, unchanged data values to be stored due to handling all data cleaning, manipulation, encoding, etc

APPLICATION FUNCTIONALITY– Background Processes: Distributions

Age Distribution from Training Data



Age Distribution from 'New' Data

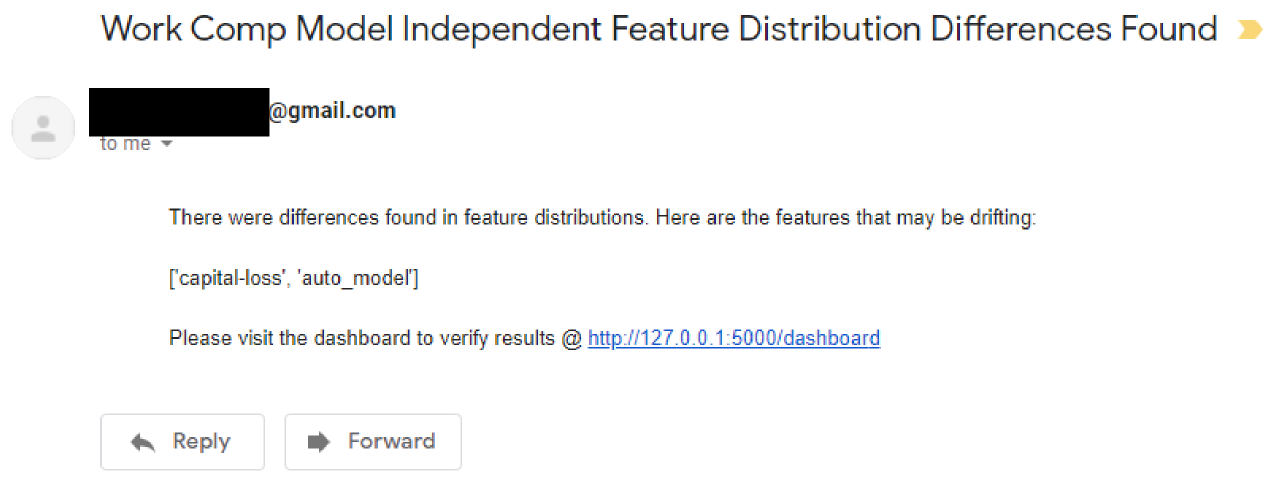


- **Compare Feature Distributions Through-Time to Detect Drift**
  - Models tend to be less predictive as data input changes
  - Process to compare new data to training data
- **Mann-Whitney Test**
  - Non-parametric statistical test to detect difference in distribution
  - Allowed for several different data types

*Mann-Whitney Test Result:
< .05 = Statistically significant (Difference in distribution)
> .05 = Statistically not significant (No difference in distribution)

APPLICATION FUNCTIONALITY– Background Processes: Email

Generated Email:


Work Comp Model Independent Feature Distribution Differences Found

[redacted]@gmail.com
to me

There were differences found in feature distributions. Here are the features that may be drifting:

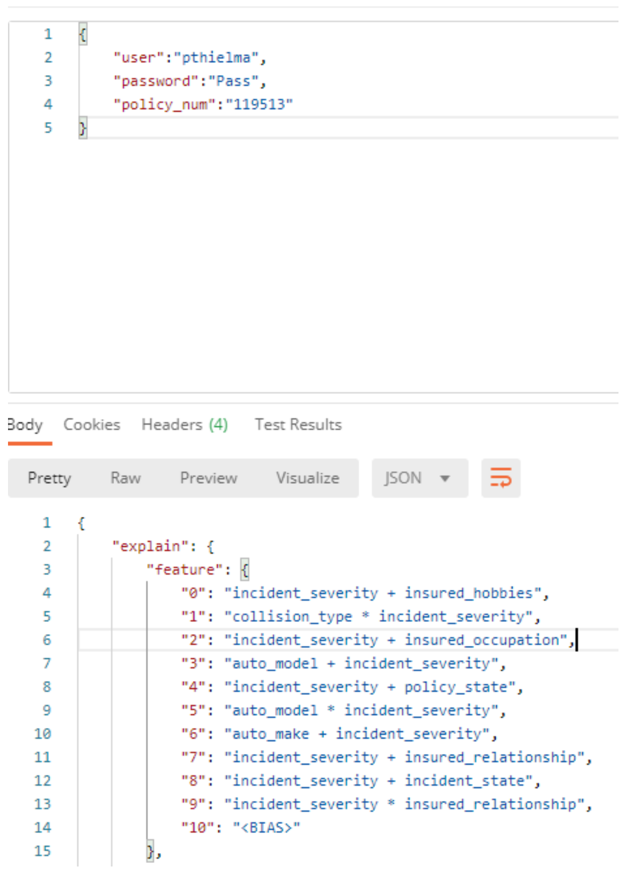
['capital-loss', 'auto_model']

Please visit the dashboard to verify results @ http://127.0.0.1:5000/dashboard

- **Creates SMTP Email Server**
  - Simple Mail Transfer Protocol
  - Server application to send / receive email
- **Automated Alert for Distribution Changes**
  - Provide an automated method for notification
  - Identify what features may be drifting
  - Runs comparison every 24 hours using rolling window
  - Provide a link to interactive dashboard to verify

APPLICATION FUNCTIONALITY– Score Function

JSON Input / Return



```
{
    "user":"pthielma",
    "password":"Pass",
    "policy_num":"119513"
}
```

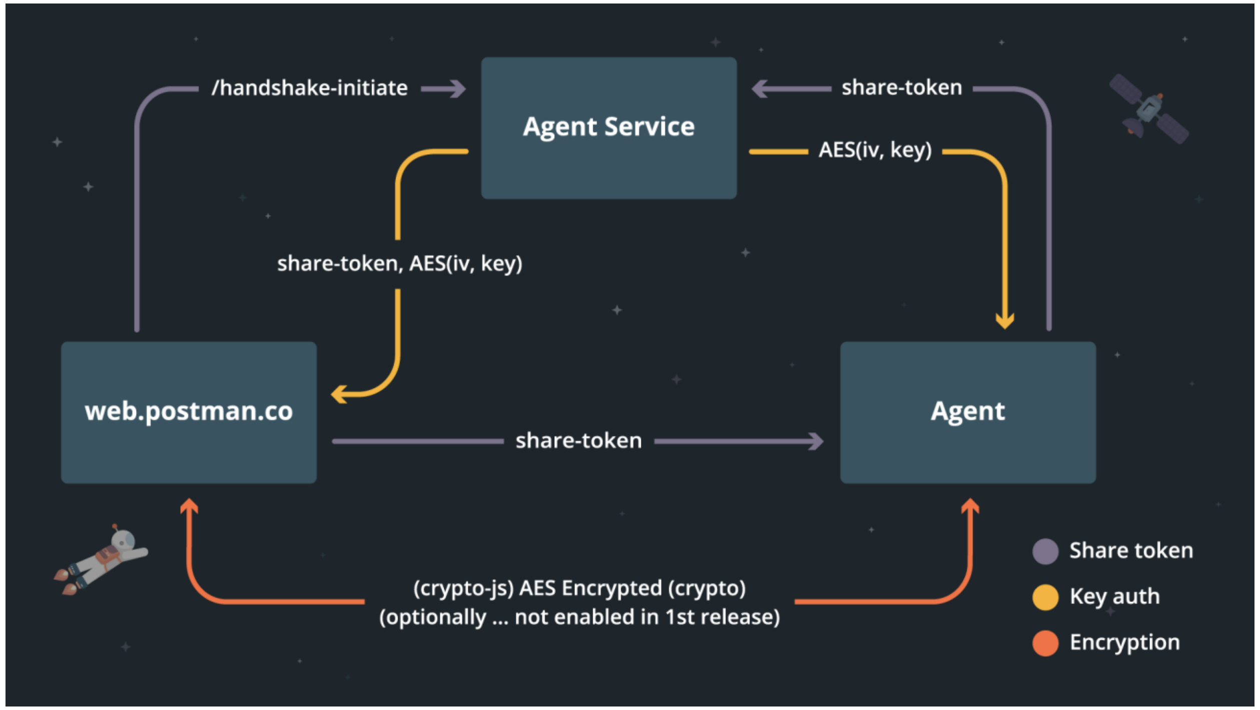
```
{
    "explain": {
        "feature": {
            "0": "incident_severity + insured_hobbies",
            "1": "collision_type * incident_severity",
            "2": "incident_severity + insured_occupation",
            "3": "auto_model + incident_severity",
            "4": "incident_severity + policy_state",
            "5": "auto_model * incident_severity",
            "6": "auto_make + incident_severity",
            "7": "incident_severity + insured_relationship",
            "8": "incident_severity + incident_state",
            "9": "incident_severity * insured_relationship",
            "10": "<BIAS>"
        },
```

What it does

- Accepts policy number ID as JSON input
- Makes database call to gather needed data
- Preprocesses data using functions created during modeling
- Creates interactive features from Feature Tools
- Calls model for prediction
- Runs ELI5 for prediction explain
- Stores scoring run result in database
- Returns results as JSON

APPLICATION FUNCTIONALITY– Score Function: Postman

Postman

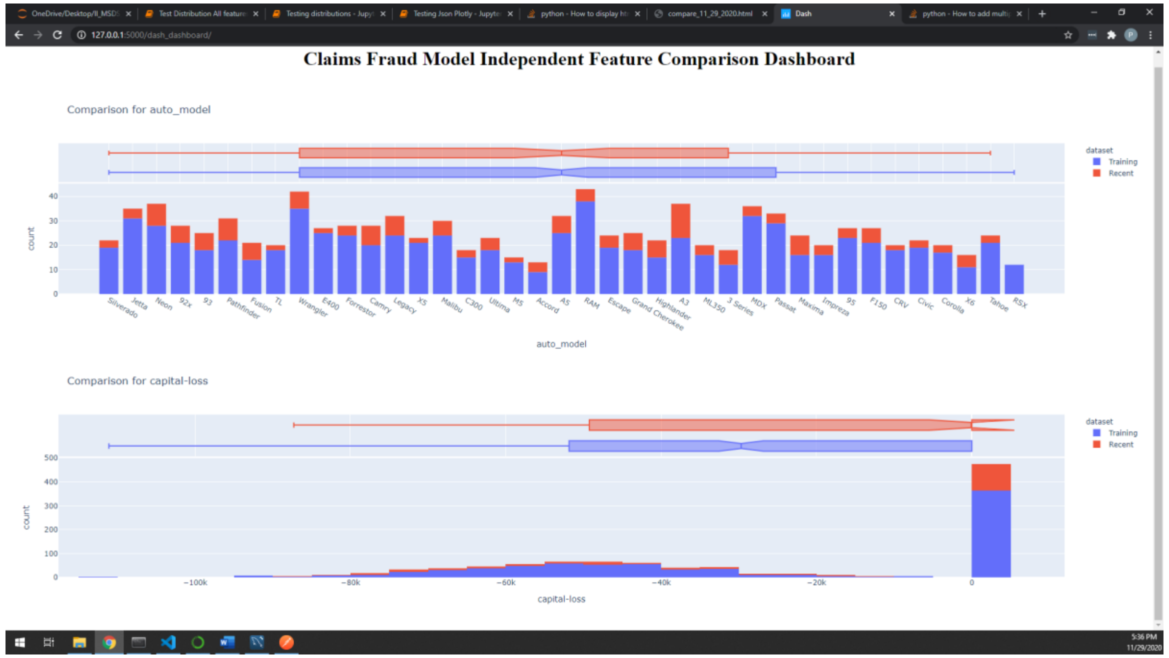


(Lane, 2020)

- **API client**
  - Emulates an application call to our API
  - Test our application for various inputs
  - Flexible environment to set up API call collections
- **Different HTTP actions available**
  - GET, POST, PUT, COPY, etc
- **Only used for testing!**
  - Actual API calls built into adjuster application
  - Helpful in determining JSON format

APPLICATION FUNCTIONALITY– Dashboard

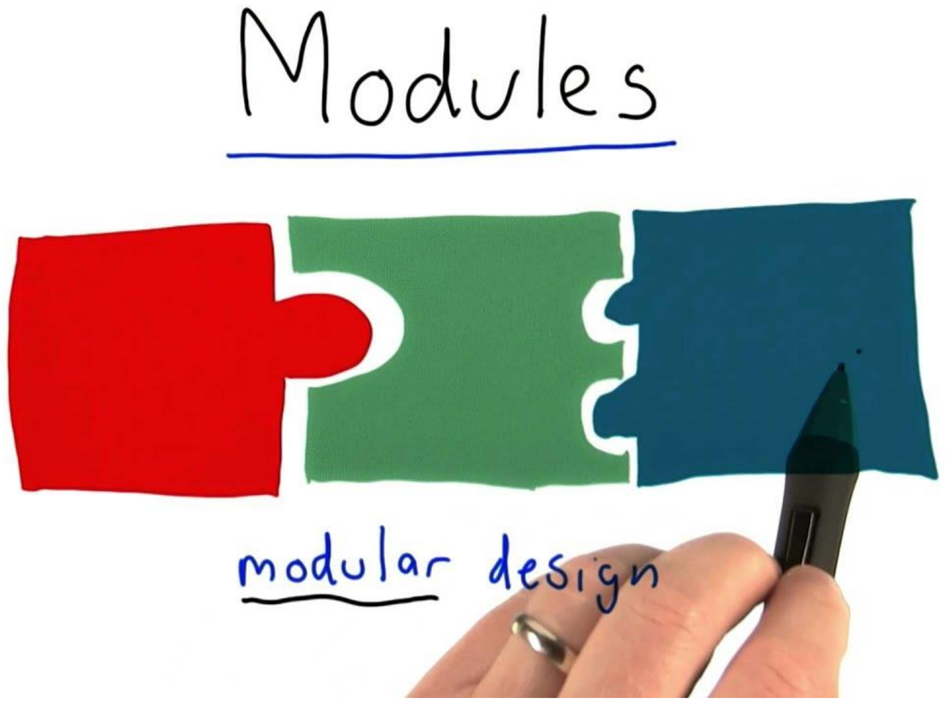
Framework in Place to Add Insights as Needed


127.0.0.1:5000/dash_dashboard/
Claims Fraud Model Independent Feature Comparison Dashboard
Comparison for auto_model
dataset
Training
Recent
count
auto_model
Comparison for capital-loss
capital-loss

- **Simply click of link from email**
  - Just like any other website, add URL to browser and go
  - Test our application for various inputs
  - Flexible environment to set up API call collections
- **Web app better than report**
  - Ability for everyone in organization to view in real-time
  - Get to what is important to you!
- **Only used for testing!**
  - Actual API calls built into adjuster application
  - Helpful in determining JSON format

CONTENTS

TEAM IMPACT– Modularity and Integration



(Innovation, 2015)

Provides framework for future products

- Re-use components of application for other projects
- Don’t start from scratch each time!

Easily integrate with other IT applications

- Uses industry standard methodology
- Deploy in different environments
- Python provides limitless capabilities compared to other implementation methods

TEAM IMPACT– Model Re-calibration Efficiency / Advanced Insights

(API, n.d.)

- **Models get stale!**
  - Re-calibrate based on model or feature performance degradation instead of waiting
  - Move from development to deployment with a simple code promote, rather than recode
- **Continuous learning**
  - Candidate model constantly training and running behind the scenes
  - Compare existing production model to new modeling methods
- **Dashboard benefits**
  - Allow to explore performance with ease
  - Ability to add important metrics

TEAM IMPACT– Faster Model Deployment

(Finger, 2014)

- **Quickly develop and deploy**
  - Offer customer more products due to speed of implementation
  - Reduce turnaround time between customer interaction. Where were we at with this project again? Oh! Its already implemented!
- **Better testing**
  - Allow for team testing of production model, rather than assist other teams that need to implement

CONTENTS

BUSINESS IMPACT– Efficiency

(Rodnitzky, 2016)

- **Under Promise, Over Deliver**
  - Reduce work hours for product development
  - Deliver quality products at an increased rate
- **Focus on what is important**
  - Spend valuable time on quality work (modeling, data exploration), rather than deployment logistics

BUSINESS IMPACT– Input From User

(I Stock, n.d.)

- **Improve presentations with hands on demo**
  - Put a demo product in the hands of user, rather than a final product months later
  - Provides important feedback from others:
    - “I am colorblind and didn’t even see that button change!”
    - “If I reload with this setting, it gives me an error?”
- **Faster feedback loop**
  - Create test applications for presentation when pitching a new product idea
  - Give the user a better understanding of what they are receiving, before its final

BUSINESS IMPACT– More Products: The Sky is the Limit!

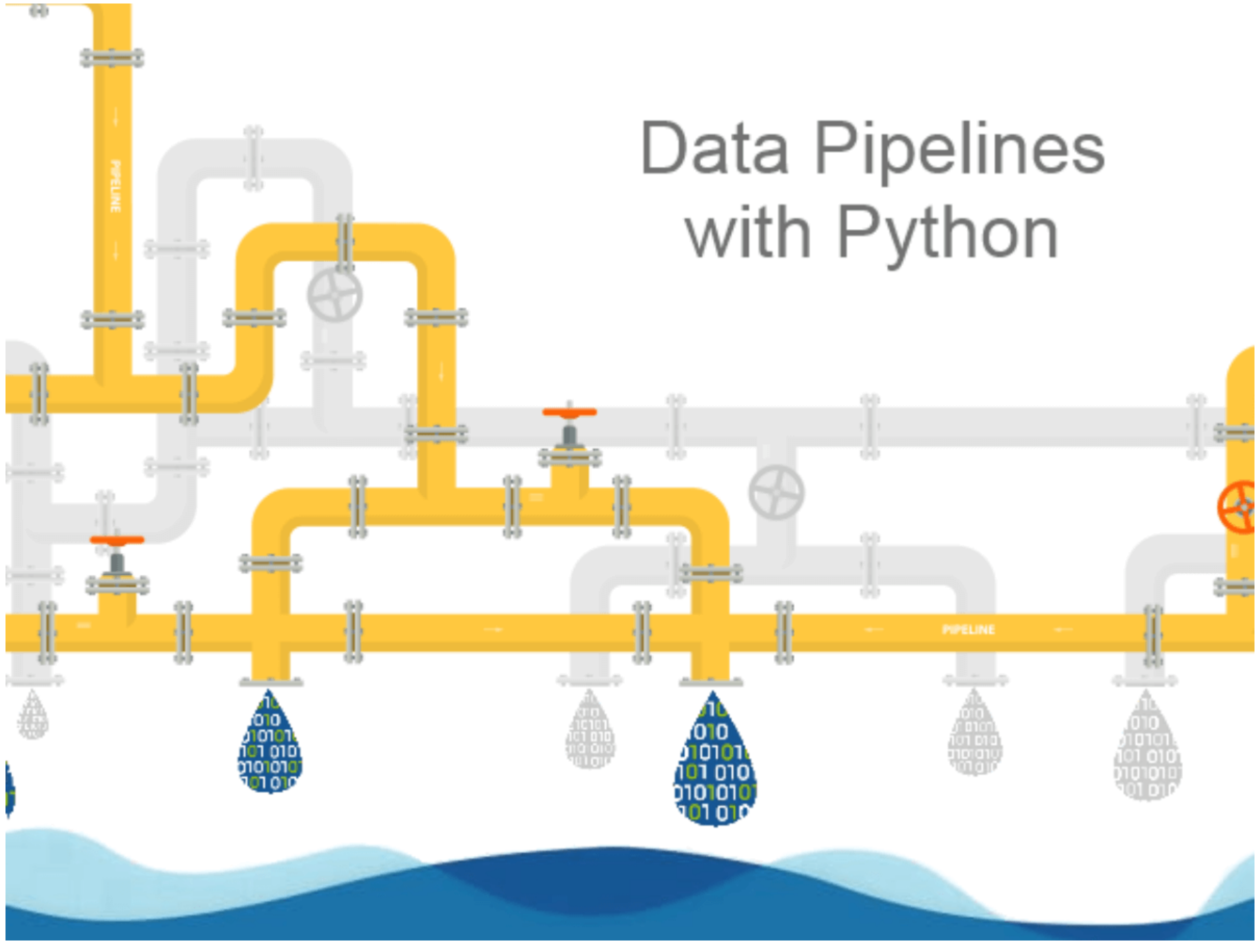
Ability to request different product types (Automation?)

Imagine a business process in which each month an employee receives a report of the current pricing for a product the company sells. This employee then searches the top 5 competitors who also sell this product. They then create a spreadsheet with the differences in product price between the 6 of them. They then calculate a proposed price change based on past data and sends the report to a manager. The manager reviews it and so on.

Even some of the seemingly easy tasks can be automated and built into a deployed app such as this. With the leverage of data scientists, statisticians, and other data savvy individuals, application development can be taken to the next level.

CONTENTS

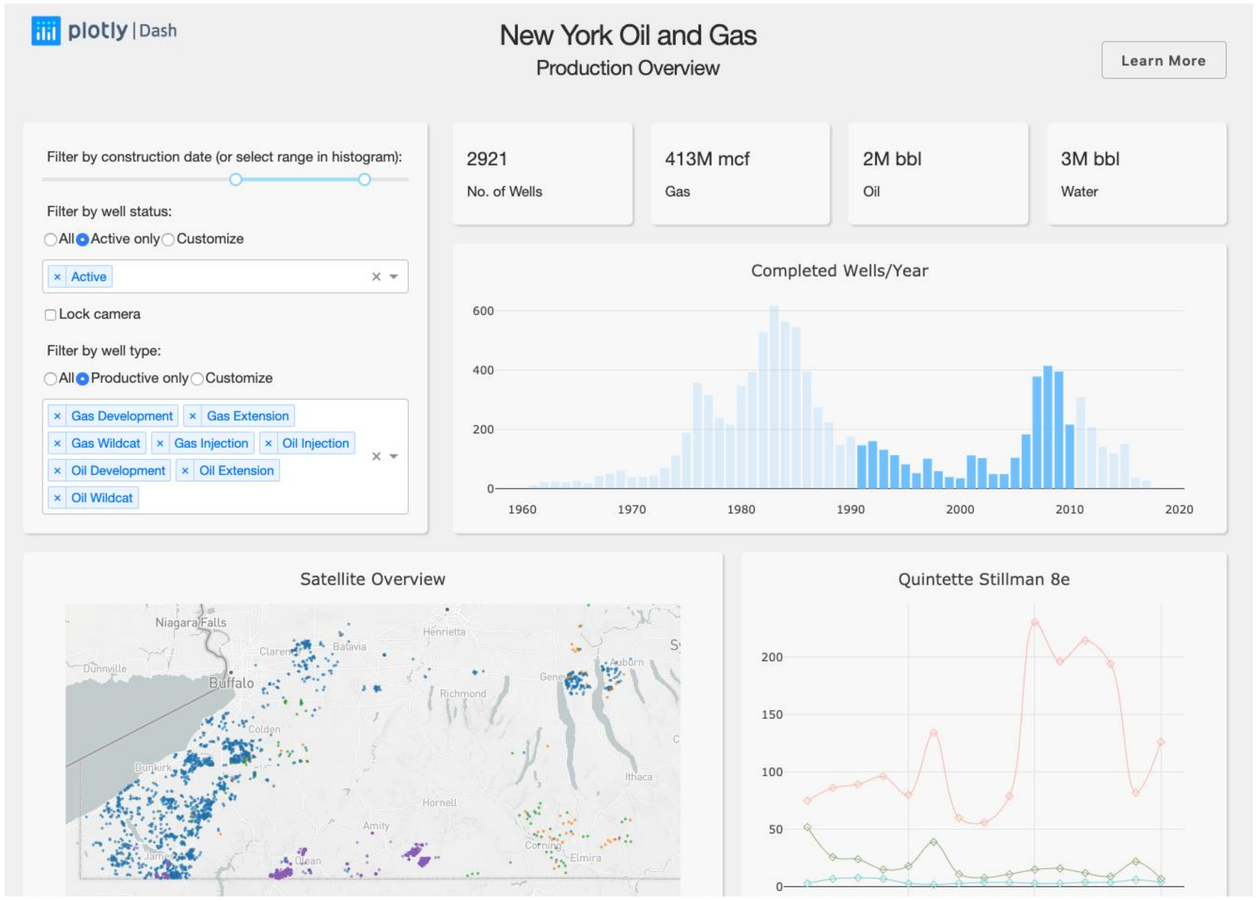
NEXT STEPS – SQL Replication / Data Pipelining



(Bohorquez, 2020)

- **Maintain single source integrity**
  - Data streaming from transactional
  - Data always up-to-date and consistent
- **Development queries straight to production**
  - No need to rewrite data mining against a separate database
- **Realtime analytics**
  - Offer products that use the most current data

NEXT STEPS – Dashboard


plotly | Dash

New York Oil and Gas
Production Overview

Learn More

Filter by construction date (or select range in histogram):

Filter by well status:
All [✓ Active only] Customize

× Active

Lock camera

Filter by well type:
All [✓ Productive only] Customize

× Gas Development × Gas Extension × Gas Wildcat × Gas Injection × Oil Injection × Oil Development × Oil Extension × Oil Wildcat

2921
No. of Wells

413M mcf
Gas

2M bbl
Oil

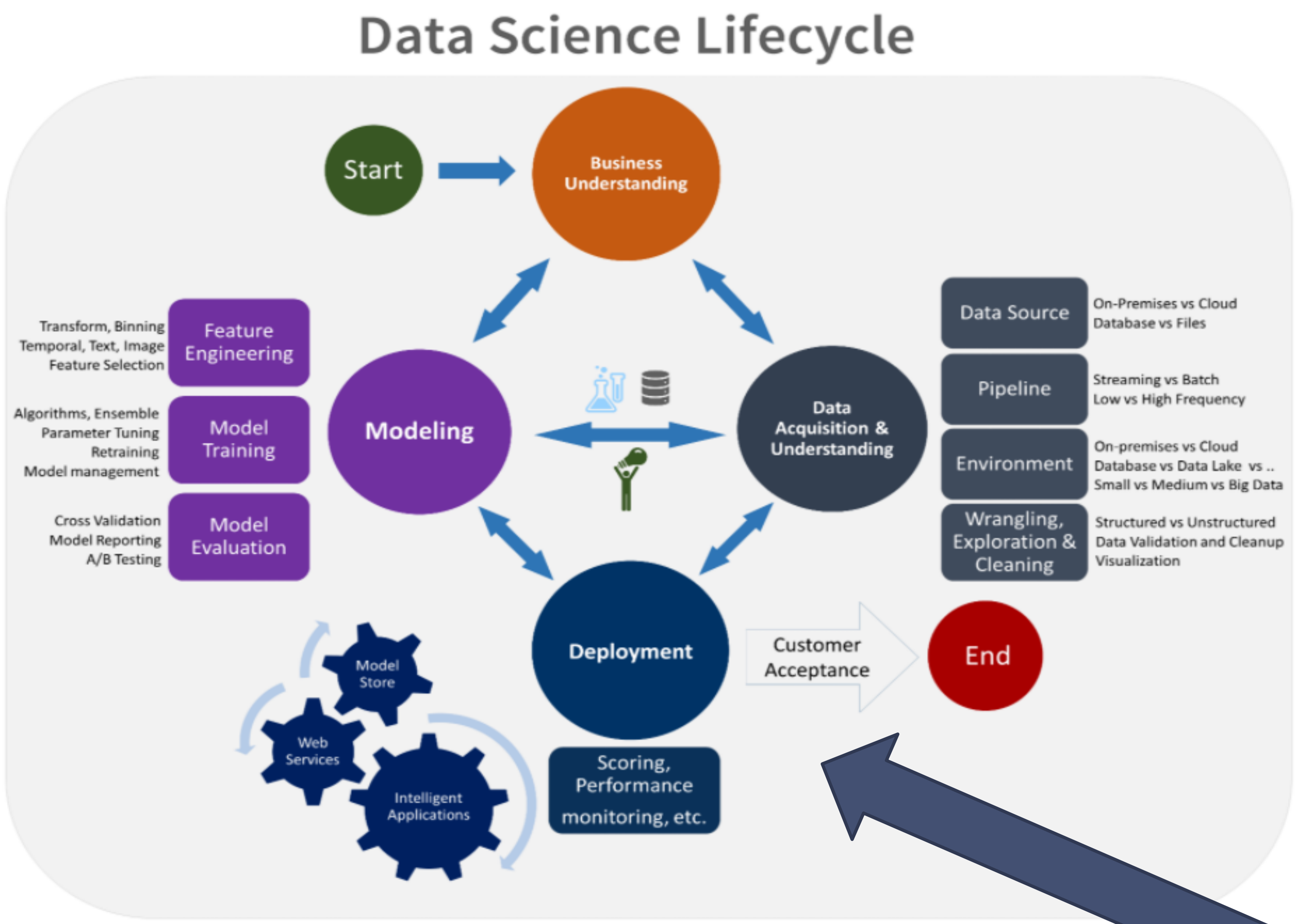
3M bbl
Water

(Dash, n.d.)

- **Web-app like dashboard**
  - Add additional dropdowns and screens
  - Additional filter criteria
- **Add continuous learning**
  - Ability to view additional test model comparisons

CONTENTS

SUMMARY– What Did I All See Again?



(Microsoft, 2020)

Deployment and Implementation

- Model serialization
- Scoring function
- Realtime data preprocessing
- Model performance evaluation
- Realtime Dashboard
- Background Scheduling
- Distribution Drift Testing
- Email Generation

Practicum II Focus

CONTENTS

References

Anderson, K. (2017, September 7). *A Confusion of Journals - What is PubMed Now?* Retrieved from The Scholarly Kitchen: https://scholarlykitchen.sspnet.org/2017/09/07/confusion-journals-pubmed-now/

API. (n.d.). *Machine & Robotic Calibration Solutions*. Retrieved from API : https://apimetrology.com/calibration/

Bohorquez, N. (2020, March 12). *How To Create Scalable Data Pipelines With Python*. Retrieved from Active State: https://www.activestate.com/blog/how-to-create-scalable-data-pipelines-with-python/

Dash. (n.d.). *Dash Enterprise App Gallery*. Retrieved from Plotly: https://dash-gallery.plotly.host/Portal/

Discovery Place Nature. (2020, April 14). *Binary Bracelets*. Retrieved from Discovery Place Nature: https://nature.discoveryplace.org/stay-at-home-science/binary-bracelets

Eising, P. (2017, December 7). *What exactly is an API?* Retrieved from Medium: https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f

Finger, L. (2014, August 19). *3 Data Products You Need To Know*. Retrieved from Forbes: https://www.forbes.com/sites/lutzfinger/2014/08/19/3-data-products-you-need-to-know/?sh=4dc60a1866f6

I Stock. (n.d.). *Business*. Retrieved from I Stock: https://www.istockphoto.com/photos/business?phrase=business&sort=mostpopular

Inovation, S. (2015, May 3). *Modular Design*. Retrieved from YouTube: https://www.youtube.com/watch?v=DuK4dKH0zOA

Lane, K. (2020, July 16). *Introducing the Postman Agent: Send API Requests from Your Browser without Limits*. Retrieved from Postman: https://blog.postman.com/introducing-the-postman-agent-send-api-requests-from-your-browser-without-limits/

Microsoft. (2020, January 10). *The Team Data Science Process Lifecycle*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle

Patient Advocate Foundation. (n.d.). *The Language of Insurance*. Retrieved from Patient Advocated: https://www.patientadvocate.org/explore-our-resources/the-language-of-insurance/

Rodnitzky, D. (2016, May 31). *Efficiency and arbitrage: two strategies to own performance marketing*. Retrieved from Marketing Land: https://marketingland.com/efficiency-arbitrage-two-strategies-performance-marketing-178328

Wikipedia. (n.d.). *Flask (web framework)*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Flask_(web_framework)