CMP6230 Draft Pipeline

Lewis Higgins - Student ID 22133848

Contents

Candidate Data Sources

For the first stage of the pipeline, data ingestion, three data sources will be identified in order to find the one that would be most optimal for the production and deployment of a machine learning model to complete a supervised learning task.

1.1 Candidate 1 - Indian Liver Patient Dataset

This dataset (Bendi Ramana and N. Venkateswarlu, 2022) consists of real data sourced from hospitals northeast of Andhra Pradesh in India. It was obtained from the UCI Machine Learning Repository, and has been previously used by Straw and Wu (2022) in their analysis of sex-related bias in supervised learning models. The UCI ML Repository is a popular host of datasets used by students, educators and researchers worldwide for machine learning (UCI Machine Learning Repository, 2024), and hosts these datasets on the cloud for public download and usage, as long as credit is given.

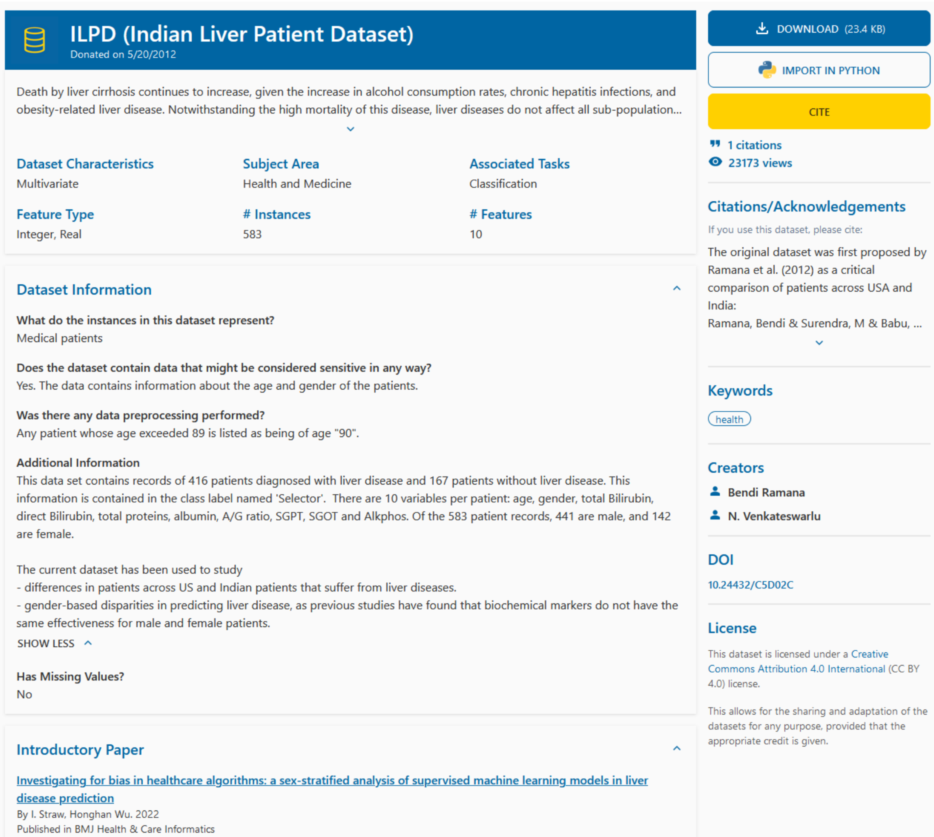

Figure 1.1: A snapshot of the dataset's UCI repository page.

This dataset in particular aims to assist in the diagnosis of liver disease due to increasing mortality rates from conditions like liver cirrhosis, and contains 584 records with 10 features as well as the "Selector" classification column, where those wihout liver disease are classed as 1, and those with liver disease are classed as 2. For the purposes of the ML model, these can be changed to 0 and 1 respectively. The dataset is a single flat-file Comma-Seperated

Values (CSV) file, which stores data by seperating each column with commas and each row with line breaks. This CSV file uses a One Big Table (OBT) schema, as seen in the entity relationship diagram in Figure 1.2, wherein all of the data within this dataset is stored in a single table. Descriptions of the columns in the dataset, as well as the associated data types, can be found in Table 1.1.

Indian Liver Patient Dataset	
Age	integer
Gender	varchar
TB	numeric
DB	numeric
Alkphos	integer
Sgpt	integer
Sgot	integer
TP	numeric
ALB	numeric
A/G Ratio	numeric
Selector	integer

Figure 1.2: An entity relationship diagram of the Indian Liver Patient Dataset.

A minor issue with this file is that it has no headers in its CSV file, meaning that when imported, Pandas will interpret the first row of data as the names of the columns, though this can be combated by adding the "names" argument when calling Pandas' "read_csv" function, seen below in Figure 1.3a.

```
df = pd.read_csv("Data/ilpd.csv")
```
✓ 0.0s

```
df.head(10)
```
✓ 0.0s

	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.9	1
0	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
1	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
2	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
3	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
4	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1
5	26	Female	0.9	0.2	154	16	12	7.0	3.5	1.00	1
6	29	Female	0.9	0.3	202	14	11	6.7	3.6	1.10	1
7	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	2
8	55	Male	0.7	0.2	290	53	58	6.8	3.4	1.00	1
9	57	Male	0.6	0.1	210	51	59	5.9	2.7	0.80	1

(a) Importing without supplying column names.

```
df = pd.read_csv("Data/ilpd.csv",
                 names = ["Age", "Gender", "TB", "DB", "Alkphos", "Sgpt", "Sgot", "TP", "ALB", "AGRatio", "Selector"])
```
✓ 0.0s

```
df.head(10)
```
✓ 0.0s

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AGRatio	Selector
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1
6	26	Female	0.9	0.2	154	16	12	7.0	3.5	1.00	1
7	29	Female	0.9	0.3	202	14	11	6.7	3.6	1.10	1
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	2
9	55	Male	0.7	0.2	290	53	58	6.8	3.4	1.00	1

(b) Importing with the column names.

Figure 1.3: Importing the erroneous CSV using Pandas. The column headers are highlighted in a red box.

A preliminary analysis of the file to ascertain the data types of each column, seen in Figure 1.4, also revealed that there were 4 missing values in the A/G ratio column. It is possible that these missing values could be imputed rather than deleted, as it may be possible to calculate what the A/G ratio of these rows would have been in the Data Preprocessing stage of a pipeline.

```
df.dtypes
```

```
Age           int64
Gender       object
TB          float64
DB          float64
Alkphos       int64
Sgpt          int64
Sgot          int64
TP          float64
ALB         float64
AGRatio     float64
Selector      int64
```

Figure 1.4: The data types of the Indian Liver Patient Dataset.

```
df.isna().sum()
```

```
Age         0
Gender      0
TB          0
DB          0
Alkphos     0
Sgpt        0
Sgot        0
TP          0
ALB         0
AGRatio     4
Selector    0
```

(a) Four missing values are identified.

```
df[df["AGRatio"].isna()]
```

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AGRatio	Selector
209	45	Female	0.9	0.3	189	23	33	6.6	3.9	NaN	1
241	51	Male	0.8	0.2	230	24	46	6.5	3.1	NaN	1
253	35	Female	0.6	0.2	180	12	15	5.2	2.7	NaN	2
312	27	Male	1.3	0.6	106	25	54	8.5	4.8	NaN	2

(b) The four rows in question.

Figure 1.5: The identification of four missing values in the A/G ratio column.

Column	Description
Age	The patient's age. **Ages of 90 or over were listed as 90 before this dataset was published.**
Gender	The patient's gender, either "Male" or "Female".
TB	Total bilirubin. Bilirubin is a substance produced by the liver, and a high presence of it may be indicative of liver problems (Mayo Clinic, 2024).
DB	Direct bilirubin. This is a slightly different form of bilirubin that is formed after the liver has processed it.
Alkphos	Levels of alkaline phosphate - an enzyme in the body produced by the liver. Too much may indicate liver disease. (Cleveland Clinic, 2024)
Sgpt	Another enzyme found in the liver, where too much can indicate liver problems.
Sgot	Levels of AST in the blood, where too much indicates liver problems.
TP	Total proteins.
ALB	Albumin - a protein in blood plasma. Too little of this may indicate liver problems.
A/G Ratio	The ratio of albumin to globulin, which is another blood protein.
Selector	The classifier, indicating if the person has liver disease. The target column for the ML model.

Table 1.1: The descriptions of each column in the Indian Liver Patient Dataset.

This dataset can be used to develop a supervised machine learning model for binary classification using the ten predictor variables and the ground truth Selector column, which will be used in measuring the accuracy of the model. There is a clear positive purpose for developing such a model; as previously mentioned, mortality rates from liver disease are high, and an early diagnosis that could leverage the power of machine learning can greatly enhance the odds of successful treatment.

1.2 Candidate 2 - Loan Approval Classification Dataset

This dataset was sourced from Kaggle's cloud servers under an Apache 2.0 license, which states that the dataset can be used as long as credit is given to the original author, and takes the form of a flat-file CSV using a One Big Table schema.

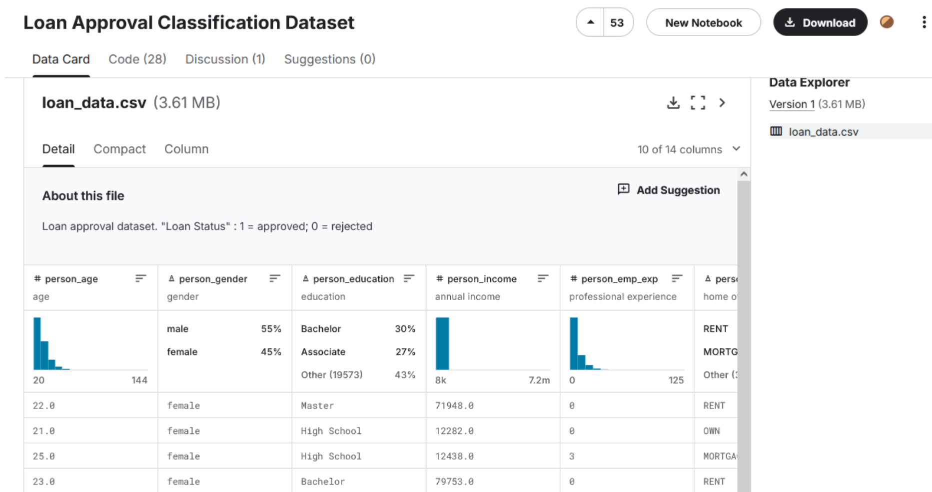

Figure 1.6: A snapshot of the Loan dataset's Kaggle page.

Unlike Candidate 1, this dataset does not consist of real data, and instead consists of synthetic data. This is likely due to the fact that this dataset, if it used real data, would contain extremely personal information that could not be shared online due to legislation such as GDPR. This particular dataset is an enhanced version of a different credit risk dataset, which also did not provide an original source and is presumably synthetic data. The dataset consists of 45,000 records and 14 features, with one of these being the ground truth target variable "loan_status", which is whether the person should be given a loan or not. As such, it is well suited for a binary classification model, using the first 13 features as predictor variables. This can also be observed from the 28 notebooks on Kaggle that utilise this dataset. The data types for each column can be seen in the entity relationship diagram and Pandas code in Figures 1.7 and 1.4, and descriptions of each column can be seen in Table 1.2.

Loan Approval Classification Dataset	
person_age	numeric
person_gender	varchar
person_education	varchar
person_income	numeric
person_emp_exp	integer
person_home_ownership	varchar
loan_amnt	numeric
loan_intent	varchar
loan_int_rate	numeric
loan_percent_income	numeric
cb_person_cred_hist_length	numeric
credit_score	integer
previous_loan_defaults_on_file	varchar
loan_status	integer

Figure 1.7: An entity relationship diagram of the Loan Approval Classification Dataset.

```
df_loan.dtypes

person_age                          float64
person_gender                        object
person_education                     object
person_income                       float64
person_emp_exp                        int64
person_home_ownership                object
loan_amnt                           float64
loan_intent                          object
loan_int_rate                       float64
loan_percent_income                 float64
cb_person_cred_hist_length          float64
credit_score                          int64
previous_loan_defaults_on_file       object
loan_status                           int64
```

Figure 1.8: The data types of the Loan Approval Classification Dataset.

```
df_loan.isna().sum()

person_age                        0
person_gender                     0
person_education                  0
person_income                     0
person_emp_exp                    0
person_home_ownership             0
loan_amnt                         0
loan_intent                       0
loan_int_rate                     0
loan_percent_income               0
cb_person_cred_hist_length        0
credit_score                      0
previous_loan_defaults_on_file    0
loan_status                       0
```

Figure 1.9: No missing values in the dataset.

Column	Description
person_age	The age of the person.
person_gender	The person's gender.
person_education	The person's highest level of education.
person_emp_exp	The person's years of employment experience.
person_home_ownership	Home ownership status (for example rent, own, mortgage)
loan_amnt	The amount of money requested.
loan_intent	The purpose of the loan.
loan_int_rate	The interest rate of the loan.
loan_percent_income	Loan amount as a percentage of the person's yearly income.
cb_person_cred_hist_length	Length of credit history in years.
credit_score	Credit score of the person.
previous_loan_defaults_on_file	If the person has defaulted on a loan before.
loan_status	Whether the loan should be approved. 1 if yes, 0 if no.

Table 1.2: The descriptions of each column in the dataset.

1.3 Candidate 3 - Spotify Likes Dataset

This dataset was sourced from Kaggle, a platform similar to the UCI ML repository in its purpose for students and researchers that acts as a search engine for datasets, but also allows its users to host competitions, upload their machine learning models, and also upload their own Python notebooks. This dataset is stored on their servers on the cloud, and is free to download and use.

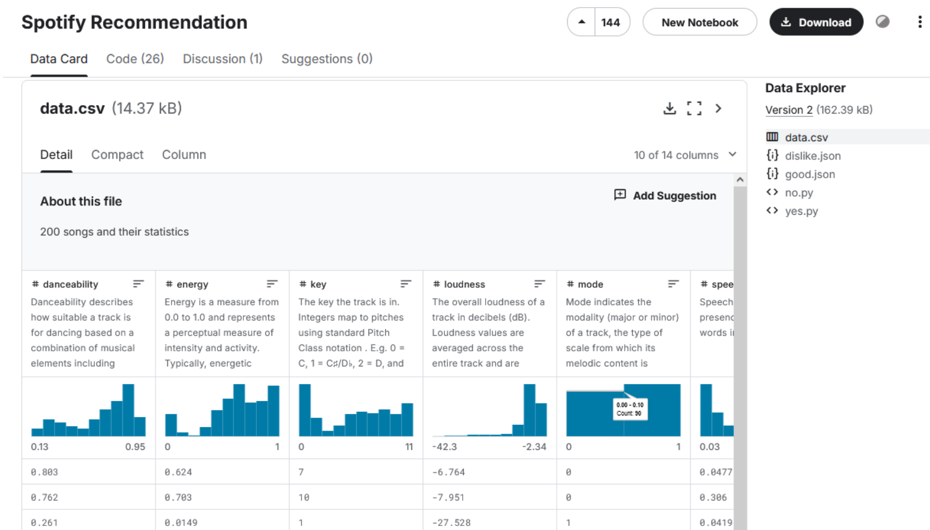

Figure 1.10: A snapshot of the Spotify dataset's Kaggle page.

The data itself is split over two JavaScript Object Notation (JSON) files, but also fully present in an included CSV file, with all three utilising a One Big Table schema. The download also includes two Python files, which have the JSON data stored in Python dictionaries for ease of access, though these will not be used in this brief analysis. JSON files store data in **key-value pairs**, such as in the example snippet of this dataset depicted in Figure 1.11.

```
"audio_features": [
  {
    "danceability": 0.357,
    "energy": 0.98,
    "key": 6,
    "loudness": -6.835,
    "mode": 1,
    "speechiness": 0.079,
    "acousticness": 0.0000522,
    "instrumentalness": 0.843,
    "liveness": 0.0768,
    "valence": 0.368,
    "tempo": 96.969,
    "type": "audio_features",
    "id": "4pFC6tuWErxbO61oFFq3BQ",
    "uri": "spotify:track:4pFC6tuWErxbO61oFFq3BQ",
    "track_href": "https://api.spotify.com/v1/tracks/4pFC6tuWErxbO61oFFq3BQ",
    "analysis_url": "https://api.spotify.com/v1/audio-analysis/4pFC6tuWErxbO61oFFq3BQ",
    "duration_ms": 242760,
    "time_signature": 4
  },
```


Figure 1.11: A snippet of the JSON data, viewed in Visual Studio Code.

Every row in the JSON files is part of the single "audio_features" key, and each new row is seperated by curly braces {}. Each column is then given as a key-value pair, such as the first row in Figure 1.11, where "danceability" is the key, and 0.352 is the associated value. This dataset does consist of real data, sourced from the author's personal liked songs directly via the Spotify API. There are 195 rows of data, with 100 liked songs, and 95 disliked songs. Liked and disliked songs are seperated into two JSON files, named "dislike" and "good". The two JSON files have 18 features, as depicted in Figure 1.12.

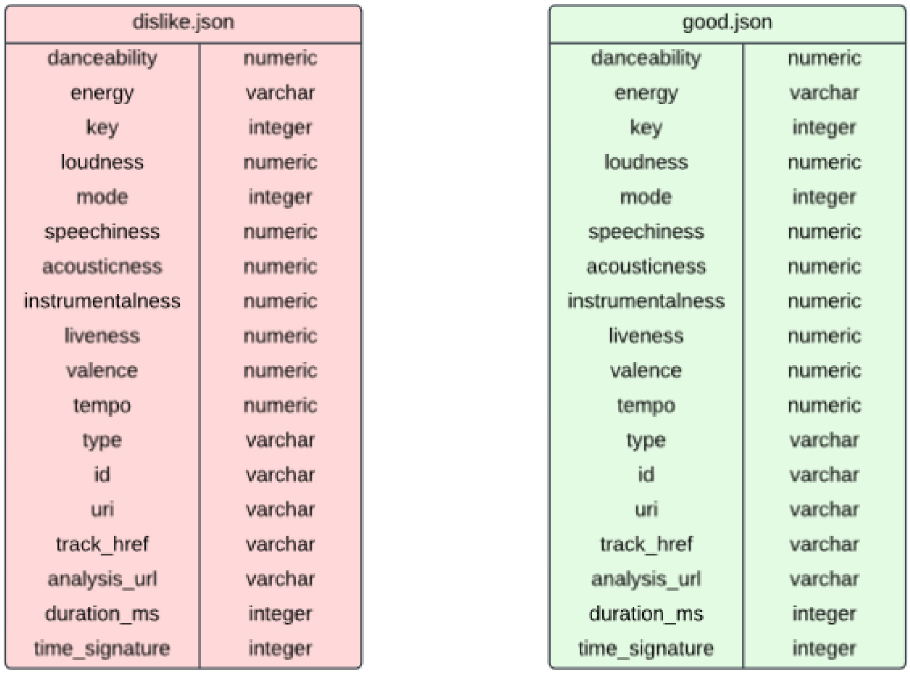

Figure 1.12: An entity relationship diagram of the two JSON files. Data does not overlap between them, so they have no relation.

This dataset has been used to create machine learning models before, most notably by its own author, who has a public Github repository showcasing their work (Brice-Vergnou, 2024). Before publicising this data, however, the author had done some preprocessing of their own, having included the additional CSV file, produced as a result of merging the two JSON files into one CSV and removing unnecessary columns, as depicted in Figure 1.13. Therefore, my preliminary Pandas analysis of the data types and missing values will only be performed on this CSV, seen in Figures 1.14 and 1.15.

data.csv	
danceability	numeric
energy	varchar
key	integer
loudness	numeric
mode	integer
speechiness	numeric
acousticness	numeric
instrumentalness	numeric
liveness	numeric
valence	numeric
tempo	numeric
duration_ms	integer
time_signature	integer
liked	integer

Figure 1.13: An entity relationship diagram of the preprocessed CSV file.

```
df_spotifyCSV.dtypes
```

```
danceability        float64
energy              float64
key                   int64
loudness            float64
mode                  int64
speechiness         float64
acousticness        float64
instrumentalness    float64
liveness            float64
valence             float64
tempo               float64
duration_ms           int64
time_signature        int64
liked                 int64
```

Figure 1.14: The data types of the Spotify Likes Dataset.

```
df_spotifyCSV.isna().sum()
```

```
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
duration_ms         0
time_signature      0
liked               0
```

Figure 1.15: No missing values in the dataset.

While a machine learning classification problem can definitely be performed on this dataset to identify if the author would like a song, it has significantly less of a positive impact than Candidates 1 and 2, as this dataset is the author's subjective belief rather than objective fact that can be applied to other people. Nevertheless, the descriptions of each column can be found in Table 1.3.

Column	Description
Danceability	How suitable a song is for dancing, calculated from the tempo, rhythm stability, beat strength and overall regularity. 1.0 means it is very danceable.
Energy	The intensity and activity of a song. For example, death metal is high energy, whereas classical music is low intensity. 1.0 is the most energetic.
Key	The musical key the song is in, converted to an integer using standard pitch class notation.(Butterfield, 2024)
Loudness	The averaged decibel volume of a song, typically between -60 and 0 dB.
Mode	Whether a song is in major or minor scale. 1 is major and 0 is minor.
Speechiness	The calculated presence of spoken words in a song.
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Instrumentalness	Whether a song has no vocals.
Liveness	Whether a live audience can be heard as part of a song.
Valence	The musical positiveness of a song.
Tempo	The beats per minute of a song.
Duration_MS	The duration of a song in milliseconds.
Time signature	The estimated time signature of the song.
Liked	The target variable, indicative of whether the author liked the song or not.
Type	Always "audio_features". Not a relevant predictor.
ID	Spotify's own unique ID for a song. Not a relevant predictor.
URI	Spotify's URI for the song. Not a relevant predictor.
Track HREF	A link to the song on Spotify's API. Not a relevant predictor.
Analysis URL	A link to the song's audio analysis data. Not a relevant predictor.

Table 1.3: The descriptions of each column in the Spotify songs dataset (Spotify, 2024). Red columns are only present in the CSV, whereas green columns are only present in the JSONs.

These measurements and the descriptions are part of Spotify's API, and are automatically calculated when songs are uploaded to the service. The ground truth of the dataset is present in the CSV file as the "liked" classifier column, and a train/test split can be implemented for predictions, which is aided by the fact that this dataset is well balanced (100 liked to 95 disliked).

1.4 Chosen dataset

Of the three candidates presented, the most suitable for a machine learning operations pipeline would be Candidate 2, the loan approval dataset. As mentioned in Section 1.2, this dataset possesses many predictor variables and an adequate amount of data to train a supervised learning classification model to classify whether an individual should be allowed a loan or not. While the data in this dataset is synthetic due to its real equivalent being highly protected under data protection legislation, the model trained from said synthetic data could be applied to real data using what it has learned, and could greatly expedite the process of loan approvals.

Planning the MLOps Pipeline

All machine learning operations (MLOps) follow a five-step repeatable pipeline, outlined in Figure 2.1, where the output of one stage becomes the input of the next. The pipeline begins with raw data and finishes with a trained machine learning model, and is often repeated at certain intervals, which could be as simple as once a day, or it could be repeated as new data becomes available. This repetition is performed automatically, so that the final model can become progressively more accurate. Because the process must be repeatable and automated, it is essential that data is validated to ensure that one run of the pipeline where the data may have been corrupted somehow would not cause issues, which would quickly spiral out of control as the pipeline is repeated again and again. These validation procedures and the software utilised for them are documented in Section 2.6. Overall, MLOps pipelines standardise the development and deployment process of machine learning models, ensuring continuous integration (CI) and continuous delivery (CD) and enhancing collaboration between data scientists and development teams.

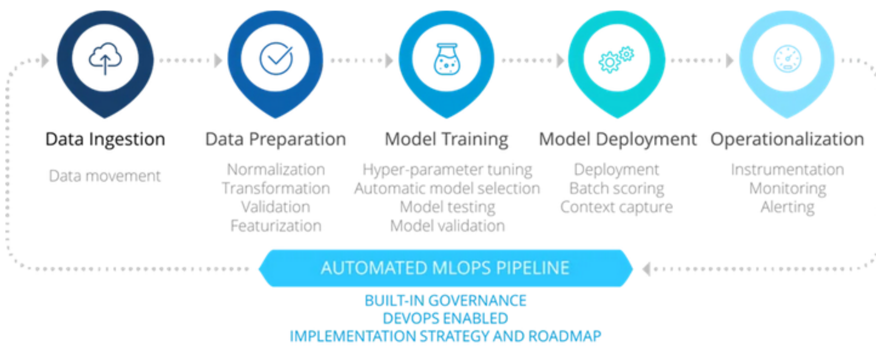

Figure 2.1: The five key steps in an MLOps pipeline (InCycle Software, 2024).

2.1 Data Ingestion

2.1.1 Description

The first step of any machine learning pipeline is data ingestion. This refers to the process of obtaining data from its original source and transferring it to a relevant storage medium, such as a database or data warehouse, to be used in later stages. This stage is undertaken by data scientists. It is of vital importance that data is not lost or corrupted when it is ingested, as this stage is the baseline for all future stages in the pipeline, and any issues here will directly impact all future stages, as previously discussed. Though, when ingesting data, it is important to understand what type of system this data will be used in, of which there are two options: Online Analytics Processing Systems (OLAP), and Online Transactional Processing Systems (OLTP), as well as how it will be loaded into this system, either using an ELT (Extract, Load, Transform) or ETL (Extract, Transform, Load) methodology.

OLAP and OLTP

OLAP	OLTP
Designed for complex queries and data analysis.	Designed for lots of short, fast queries ("transactions").
Typically store massive amounts of data, sometimes petabytes. for extremely detailed analysis.	Usually store less data for speed purposes.
Usually historical data, infrequently updated.	Typically real-time data.
Uses database schemas such as star or snowflake schema to allow for queries using many joins.	Uses normalised or denormalised models, such as One Big Table, minimising joins and maximising speed.
Slow response times, measured in minutes.	Fast response times, measured in milliseconds.

Table 2.1: A comparison of OLAP and OLTP systems (AWS, 2024b).

As mentioned in Table 2.1, OLAP systems are designed for complex and deep data analytics, which helps companies perform tasks such as analysing customer trends, while OLTP systems aim for maximum speed to complete quick transactions, which is necessary in situations like processing payments and orders.

ELT and ETL

ELT and ETL are both acronyms for the order of processes taken when ingesting data, with "Transform" either happening before or after the data is loaded into a storage medium like a data warehouse or data lake. The extract phase refers to the intiial gathering of the data from its original source, such as Kaggle for the selected dataset in this report. The transform phase refers to early modifications made to the data, such as formatting or cleaning, and the load phase refers to the transportation of the data from its original storage medium (a CSV on Kaggle's cloud servers in this case) to a more optimised and efficient system to be used in the execution of the pipeline. More details on said systems can be found in Section 2.6.

Regardless of whether ETL or ELT is used, the data is still extracted, loaded and transformed. However, the decision of which order to use can be determined from the data itself, with smaller datasets that need complex transformation being more suited to ETL, whereas larger datasets with less transformation needed are better with ELT (Smallcombe, 2024).

ELT	ETL
Data is transformed **after** being loaded into another storage medium.	Data is transformed **before** being loaded into another storage medium.
Useful if the data warehouse is a more modern system with good processing power.	Useful if the data warehouse has limited processing abilities of its own, typically seen in older systems.
Lower latency in the ingestion phase because data is immediately loaded.	Higher latency in the ingestion phase, because the data is being processed first, adding an extra time overhead where the pipeline could be held up.
Simpler to set up due to the immediate loading of the data.	Can be complex to set up and maintain, as the transformation would occur outside the warehouse.

Table 2.2: A comparison of ETL and ELT methodologies (Bartley (2024), AWS (2024a)).

It can be surmised from the analysis of each method in Table 2.2 that ETL is best applied to older systems with limited processing power, whereas ELT is better in more modern systems. It can additionally be argued that performing the ETL order of operations blurs the line between the ingestion and preprocessing stages, as the data is transformed before it is actually loaded into the system and fully ingested, whereas ELT has a clear split between the ingestion and preprocessing of the data.

2.1.2 In this project

The candidate dataset utilises the One Big Table schema, and it was previously established in Section 1.2 that it contains no missing values. The size of the dataset is the largest of the three candidates but is still small in comparison to those used in industry. The data will be ingested using ELT into a MariaDB Columnstore OLTP database for quick and efficient loading and querying. MariaDB is...

2.2 Data Preprocessing

2.2.1 Description

After the data has been ingested, the preprocessing stage begins, and is often also conducted by Data Scientists. This stage encompasses the cleaning, integration and transformation of the data in order to optimise the dataset for model development.

Cleaning refers to the identification of missing, inaccurate or malformed data within the dataset, as well as its removal or imputation where possible.

Integration is often seen in datasets with multiple tables or that have been retrieved from multiple sources, and refers to the combination of the retrieved data into a single flat file.

Transformation, also known as feature engineering, is a considerable aspect of data preprocessing which refers to the manipulation and formatting of the data, such as changing the formats of columns from numeric dates to proper date data types, as well as the handling of categorical data, such as genders, which may originally be strings. Strings cannot be interpreted by machine learning models, and therefore they are encoded into numbers using techniques such as label encoding, which converts the unique values in a column to a numerical representation, such as male being 0 and female being 1. Data is also normalised and standardised in this stage, meaning that numerical data is reduced to being between 0 and 1 to adjust the overall scale of the data. This is especially useful with algorithms such as K-Nearest Neighbours, where large differences in distance between data can mislead the classification algorithm (IBM, 2021).

Once these tasks have all been completed, the dataset will be ready to be used for model development.

2.2.2 In this project

The preprocessing in this project will utilise Python libraries including Pandas, NumPy, Matplotlib and Seaborn. These libraries are all used in data analytics processes to store the data in a format readable by Python (Pandas DataFrames), perform calculations (NumPy) and produce visualisations from the data (Matplotlib, Seaborn).

2.3 Model Development

2.3.1 Description

The model development stage uses the processed dataset from the preprocessing stage and leverages machine learning algorithms to solve the problem in question, either a classification problem where data will be identified as being of a certain category (class), or a regression problem where unknown data can be predicted. Both of these problems require the model to be "fitted" and "trained". These refer to the utilisation of the processed dataset for the recognition of patterns, associations and correlations within the data. To fit and train the data, it is split into two sets: a training set, consisting of a large majority of the data ($\tilde{8}0\%$), and a testing set which uses the remaining minority. The algorithm will then use what it has learned from the training set to make predictions on the testing set, from which the accuracy of the model can be ascertained. These processes often yield better results with larger datasets, as the algorithm will have more information to learn and make predictions based on, which is why it is preferred to not remove data from the dataset unless strictly necessary.

This stage is conducted by machine learning engineers, and its output is that of the trained machine learning model, which can then be deployed.

2.3.2 In this project

The candidate dataset poses a classification problem, and as such, the Scikit-Learn Python library will be utilised for its implementation of a Random Forest algorithm

2.4 Model Deployment

2.4.1 Description

The developed model from the previous stage of the pipeline can then be integrated into an actual environment, and can be utilised as a tool to make decisions. This stage is where software engineers will make the model available for use, and will therefore begin to be provided with unseen data, which refers to data outside of the original training dataset.

Models are typically deployed using Representational State Transfer APIs, better known as REST APIs (RedHat, 2024). REST APIs make use of typical frontend web HTTP requests (GET, POST, PUT, DELETE) from the client to give instructions to the backend machine learning model (RestfulAPI, 2023). A significant benefit of using REST APIs is the massive portability benefits provided; using a REST API means that the model can provide results to a wide variety of devices such as Windows PCs, Macs, and even phones, as anything that can make HTTP requests can interface with one.

2.4.2 In this project

The produced model will be hosted on a webserver via Uvicorn, and the REST API will be implemented via FastAPI. These two Python packages will provide an interface where the user can input the predictor variables of the dataset (age, credit score, etc.) to receive the model's output result.

2.5 Model Monitoring

2.5.1 Description

After the model is deployed, its performance is continuously monitored by data scientists. The monitoring process consists of the analysis of the model's results via metrics such as F1-score for classification models, which is calculated from the model's accuracy and precision (Kundu, 2024), or Mean Absolute Error (MAE) for regression models, which is the average of the differences between predicted and actual values. By monitoring the model, any issues with it can be quickly identified and the pipeline can be restarted to yield a higher accuracy model, which can be monitored again.

2.5.2 In this project

In this project, software such as MLFlow will be utilised to store and log information for each model iteration.

2.6 Software used in an MLOps pipeline

For each stage of the pipeline, the implementation of them for the candidate dataset was discussed with a brief overview of the software and libraries for each. This section will explore each in higher detail.

Conda, airflow, Docker, Uvicorn, FastAPI, MLFlow, MariaDB, etc.

Appendix

3.1 Data schemas

3.1.1 One Big Table (OBT)

3.1.2 Star

Dimension tables etc

3.1.3 Snowflake

Idk

Bibliography

AWS (2024a). *ETL vs ELT - Difference Between Data-Processing Approaches - AWS*. Amazon Web Services, Inc. URL: https://aws.amazon.com/compare/the-difference-between-etl-and-elt/ (visited on 11/17/2024).

AWS (2024b). *OLTP vs OLAP - Difference Between Data Processing Systems - AWS*. Amazon Web Services, Inc. URL: https://aws.amazon.com/compare/the-difference-between-olap-and-oltp/ (visited on 11/16/2024).

Bartley, Kevin (Oct. 30, 2024). *ETL vs ELT: Key Differences, Comparisons, & Use Cases*. Rivery. URL: https://rivery.io/blog/etl-vs-elt/ (visited on 11/17/2024).

Bendi Ramana and N. Venkateswarlu (2022). *ILPD (Indian Liver Patient Dataset)*. DOI: 10.24432/C5D02C.

Brice-Vergnou (Oct. 16, 2024). *Brice-Vergnou/spotify_recommendation*. original-date: 2021-07-27T10:05:51Z.

Butterfield, Sean (2024). *22b Lesson - Pitch-class integer notation*. Inquiry-Based Music Theory. URL: https://smbutterfield.github.io/ibmt17-18/22-intro-to-non-diatonic-materials/b2-tx-pcintnotation.html (visited on 11/12/2024).

Cleveland Clinic (2024). *Alkaline Phosphatase (ALP): What It Is, Causes & Treatment*. Cleveland Clinic. URL: https://my.clevelandclinic.org/health/diagnostics/22029-alkaline-phosphatase-alp (visited on 11/12/2024).

IBM (Oct. 4, 2021). *What is the k-nearest neighbors algorithm? | IBM*. URL: https://www.ibm.com/topics/knn (visited on 11/17/2024).

InCycle Software (2024). *MLOps ENTERPRISE ACCELERATOR*. URL: https://www.incyclesoftware.com/azure-machine-learning-enterprise-accelerator (visited on 11/13/2024).

Kundu, Rohit (2024). *F1 Score in Machine Learning: Intro & Calculation*. URL: https://www.v7labs.com/blog/f1-score-guide (visited on 11/17/2024).

Mayo Clinic (2024). *Bilirubin test - Mayo Clinic*. URL: https://www.mayoclinic.org/tests-procedures/bilirubin/about/pac-20393041 (visited on 11/12/2024).

RedHat (2024). *What is a REST API?* URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (visited on 11/17/2024).

RestfulAPI (Dec. 12, 2023). *What is REST?* REST API Tutorial. URL: https://restfulapi.net/ (visited on 11/17/2024).

Smallcombe, Mark (2024). *ETL vs ELT: 5 Critical Differences*. Integrate.io. URL: https://www.integrate.io/blog/etl-vs-elt/ (visited on 11/17/2024).

Spotify (2024). *Web API Reference | Spotify for Developers*. URL: https://developer.spotify.com/documentation/web-api/reference/get-audio-features (visited on 11/12/2024).

Straw, Isabel and Honghan Wu (Apr. 25, 2022). "Investigating for bias in healthcare algorithms: a sex-stratified analysis of supervised machine learning models in liver disease prediction". In: *BMJ Health & Care Informatics* 29 (1). Publisher: BMJ Publishing Group Ltd. ISSN: 2632-1009. DOI: 10.1136/bmjhci-2021-100457.

UCI Machine Learning Repository (2024). *About - UCI Machine Learning Repository*. URL: https://archive.ics.uci.edu/about (visited on 11/12/2024).