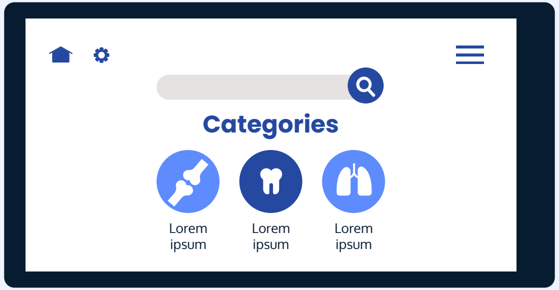
ClaimSights: Machine Learning Foresights into Health Insurance Claims

Group 5: Allan, Qudsia, Merve, and Yasmin

Table of contents

1.0

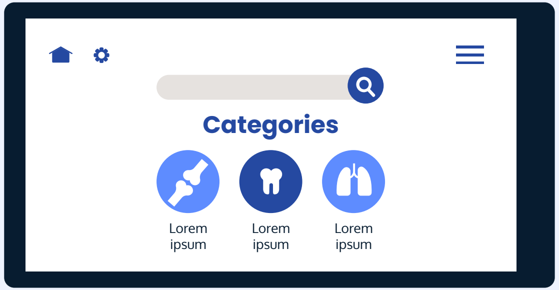
About the Data



Purpose

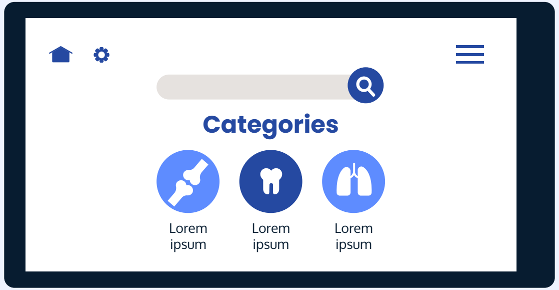
Our objective was to build a machine learning model to allow a user to forecast the likelihood to their health insurance claim claims being Rejected or Approved.

The model will aid the user in deciding if the policy of Company X suits their health insurance needs



Data source

Dataset on health insurance claims including the patient, diagnosis, and medication information and status of the claim from Company X - a health insurance company based in United Arab Emirates.



Target Audience

Educated customers looking to determine if Company X sutis their health coverage needs.

2.0

EDA, ETL and Data Preprocessing

What’s in the data

```
health_insurance_df.nunique()
✓ 0.3s

Patient                    15258
Age                          101
Sex                            2
Diagnosis_Code              1847
Diagnosis_Description       1847
Med_Code                    1612
Med_Description             1063
Quantity                      85
Status                         5
Amount_Billed               2452
Amount_Paid                 4187
dtype: int64
```

215,585

rows of

total data

2.1: EDA – Age

```
26-45    96250
46-65    71409
66+      21533
0-15     15322
16-25    11039
Name: Age_Group, dtype: int64
```

Step 1: Binnig Age groups

Range: 1 - 104 (removed errors such 127 and 134)

```python
# Bin age into groups as an additional columns
bins = [0, 15, 25, 45, 65, 104]
labels = ['0-15', '16-25', '26-45', '46-65', '66+']
health_insurance_df['Age_Group'] = pd.cut(health_insurance_df['Age'], bins=bins, labels=labels, include_lowest=True)
health_insurance_df.head()
```

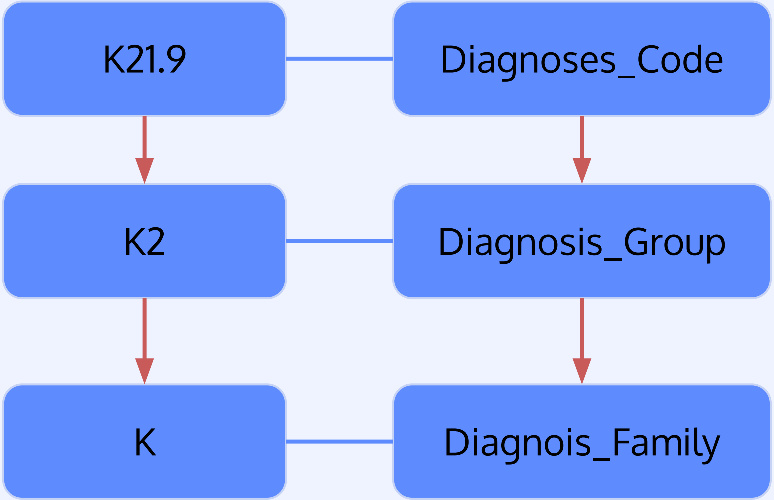
2.1: EDA – Diagnosis Codes

```
health_insurance_df.nunique()
✓ 0.3s

Patient                  15258
Age                        101
Sex                          2
Diagnosis_Code            1847
Diagnosis_Description     1847
Med_Code                  1612
Med_Description           1063
Quantity                    85
Status                       5
Amount_Billed             2452
Amount_Paid               4187
dtype: int64
```

Step 2: Grouping Diagnosis Codes

n = 1847!



```
Group: C0
Diagnosis Descriptions:
Malignant neoplasm of mouth, unspecified
```

```
Group: C1
Diagnosis Descriptions:
Malignant neoplasm of ascending colon
Malignant neoplasm of cecum
Malignant neoplasm of duodenum
Malignant neoplasm of posterior wall of hypopharynx
Malignant neoplasm of body of stomach
Malignant neoplasm of transverse colon
Malignant neoplasm of sigmoid colon
Malignant neoplasm of rectosigmoid junction
```

Diagnosis_Group = C0
Malignant Neoplasms of Digestive System and Mouth

Original Diagnoses_Group → n = 166
After grouping and merging → n = 74

2.1: EDA – Med Description

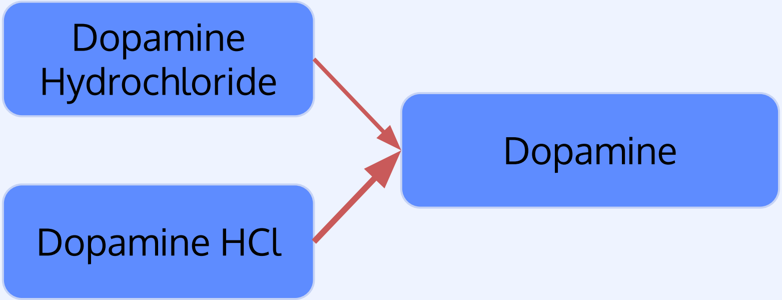
```
health_insurance_df.nunique()
✓ 0.3s

Patient                 15258
Age                       101
Sex                         2
Diagnosis_Code           1847
Diagnosis_Description    1847
Med_Code                 1612
Med_Description          1063
Quantity                   85
Status                      5
Amount_Billed            2452
Amount_Paid              4187
dtype: int64
```

Step 2: Grouping Med_Description

Med_Code > Med_Description because code is unique to medication + dose + manufacturer.

- Remove dose from description
- Group medication with the same formula



2.1: EDA – Overall

```
health_insurance_df.nunique()
✓ 0.3s

Patient                    15258
Age                          101
Sex                            2
Diagnosis_Code              1847
Diagnosis_Description       1847
Med_Code                    1612
Med_Description             1063
Quantity                      85
Status                         5
Amount_Billed               2452
Amount_Paid                 4187
dtype: int64
```

```
health_insurance_df.nunique()
✓ 0.5s

Patient                    15257
Age                           99
Age_Group                      5
Sex                            2
Diagnosis_Code              1847
Diagnosis_Group               74
Diagnosis_Family              23
Diagnosis_Description       1847
Med_Code                    1612
Med_Description             1063
Med_Description_Simp         478
Quantity                      79
Status                         2
Amount_Billed               2452
Amount_Paid                 4187
dtype: int64
```

2.2: ETL

```python
#Testing connection to database server
try:
    connection = engine.connect()
    print("Connection successful!")
    connection.close()
except Exception as e:
    print(f"Connection failed with error: {e}")
```

```
Connection successful!
```

```python
#Creating SQL Metadata to load in file
metadata = MetaData()
metadata.reflect(bind=engine)
# Get the reflected table from the metadata
reflected_employee_table = metadata.tables['claim']
```

```python
#Connecting to Database
stmt = select(reflected_employee_table)

with engine.connect() as connection:
    results = connection.execute(stmt).fetchall()
```

2.3: Data Preprocessing

```
#Changing Sexes to binary values
df['Sex'] = df['Sex'].replace({'Male': 0, 'Female': 1})
```

```
#Creating Dummies for Diagnosis_Group
diag_dummies = pd.get_dummies(df["Diagnosis_Group"])
diag_dummies.head(50)
```

```
#Scaling our data
age_scaled = StandardScaler().fit_transform(df[["Age"]])

# Diplay the first five rows of the scaled data
print(age_scaled)

#Creating a new dataframe withs scaled age
df['Age_Scaled'] = pd.DataFrame(age_scaled)
```

2.3: Data Preprocessing

Age_Scaled	Sex	A0	A1	A4	A5	A8	A9	B3	B4	...	S0	S2	S3	T1	U0	Z0	Z3	Z4	Z9	Status
-0.355374	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
-0.299123	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Rejected
0.882152	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	Paid
-0.299123	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
0.038384	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
-2.155412	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
-0.355374	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
0.150886	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
-0.917886	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	Paid
1.163408	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
1.163408	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
1.725919	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
-0.411625	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
0.432142	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
-0.524128	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid
0.994654	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	Paid
0.657147	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Paid

Target

Chosen ML Model

- Provides high accuracy
- Less prone to outliers
- Robust to overfitting
- Proficient in handling data with high dimensions
- Model performance assessed using accuracy score and classification report

4.0

Analysis

Baseline Optimizations

Initial model: **Baseline**

- Number of trees to 100

= Poor recall score for Rejected class.

Classification Report:

	precision	recall	f1-score	support
Paid	0.83	0.94	0.88	38935
Rejected	0.75	0.49	0.59	14954
accuracy			0.81	53889
macro avg	0.79	0.71	0.74	53889
weighted avg	0.81	0.81	0.80	53889

Optimizations

Optimization 1: **Balancing Class Weights**

- Number of trees to 500
- Adjusted the minimum sample split.

= This resulted in an improvement in recall for the "Rejected" class but a decrease in overall accuracy.

Classification Report:

	precision	recall	f1-score	support
Paid	0.89	0.74	0.81	38935
Rejected	0.54	0.77	0.63	14954
accuracy			0.75	53889
macro avg	0.71	0.76	0.72	53889
weighted avg	0.79	0.75	0.76	53889

Optimization 2: **Introducing SMOTE**

- SMOTE to address class imbalance

= Did not significantly differ from the previous model.

Classification Report:

	precision	recall	f1-score	support
Paid	0.89	0.74	0.81	38935
Rejected	0.53	0.77	0.63	14954
accuracy			0.75	53889
macro avg	0.71	0.76	0.72	53889
weighted avg	0.79	0.75	0.76	53889

Optimizations

Optimization 3: **SMOTE + Custom Class Weights**

- Further adjusted hyperparameters

= Increase in overall accuracy

= Better balance between recall scores for both classes.

Optimization 4: **SMOTE + Tomelinks**

- Decreased the number of trees
- Different resampling technique

= Improved balance between recall scores for both classes.

Classification Report:

	precision	recall	f1-score	support
Paid	0.87	0.82	0.84	38935
Rejected	0.59	0.68	0.63	14954
accuracy			0.78	53889
macro avg	0.73	0.75	0.74	53889
weighted avg	0.79	0.78	0.78	53889

Classification Report:

	precision	recall	f1-score	support
Paid	0.89	0.74	0.81	38935
Rejected	0.53	0.77	0.63	14954
accuracy			0.75	53889
macro avg	0.71	0.76	0.72	53889
weighted avg	0.79	0.75	0.76	53889

Random Forest

Results after optimizations:

Classification Report:

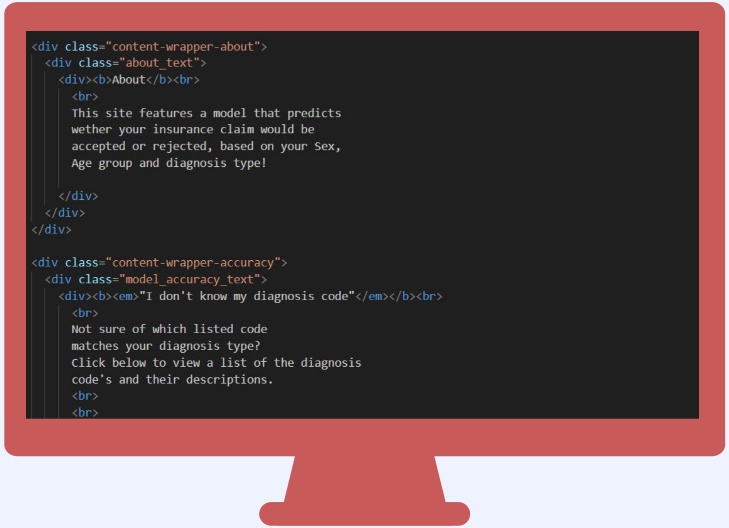
	precision	recall	f1-score	support
Paid	0.87	0.82	0.84	38935
Rejected	0.59	0.68	0.63	14954
accuracy			0.78	53889
macro avg	0.73	0.75	0.74	53889
weighted avg	0.79	0.78	0.78	53889

Accuracy: 78%

5.0

Front-end Development and Live Demonstration

5.1: Front-end Dev: Designing



```
<div class="content-wrapper-about">
  <div class="about_text">
    <div><b>About</b><br>
      <br>
      This site features a model that predicts
      wether your insurance claim would be
      accepted or rejected, based on your Sex,
      Age group and diagnosis type!

    </div>
  </div>
</div>

<div class="content-wrapper-accuracy">
  <div class="model_accuracy_text">
    <div><b><em>"I don't know my diagnosis code"</em></b><br>
      <br>
      Not sure of which listed code
      matches your diagnosis type?
      Click below to view a list of the diagnosis
      code's and their descriptions.
      <br>
      <br>
```

HTML’s, Bootstrap, CSS

- Spreading information across a second page to not overfill main page

- Making layout screen-friendly across multiple devices with <div> elements and margin styling with % values for consistency

- Enhancing user experience with pre-existing tool-tips created by bootstrap

- Utilization of buttons

5.1: Front-end Dev: Functionality

JavaScript & HTML

- Functionality of all user-interactive features done with adding event listeners, functions and conditional statements via JS

- Utilizing built-in functionality features on HTML itself such as “role=’tooltip’”

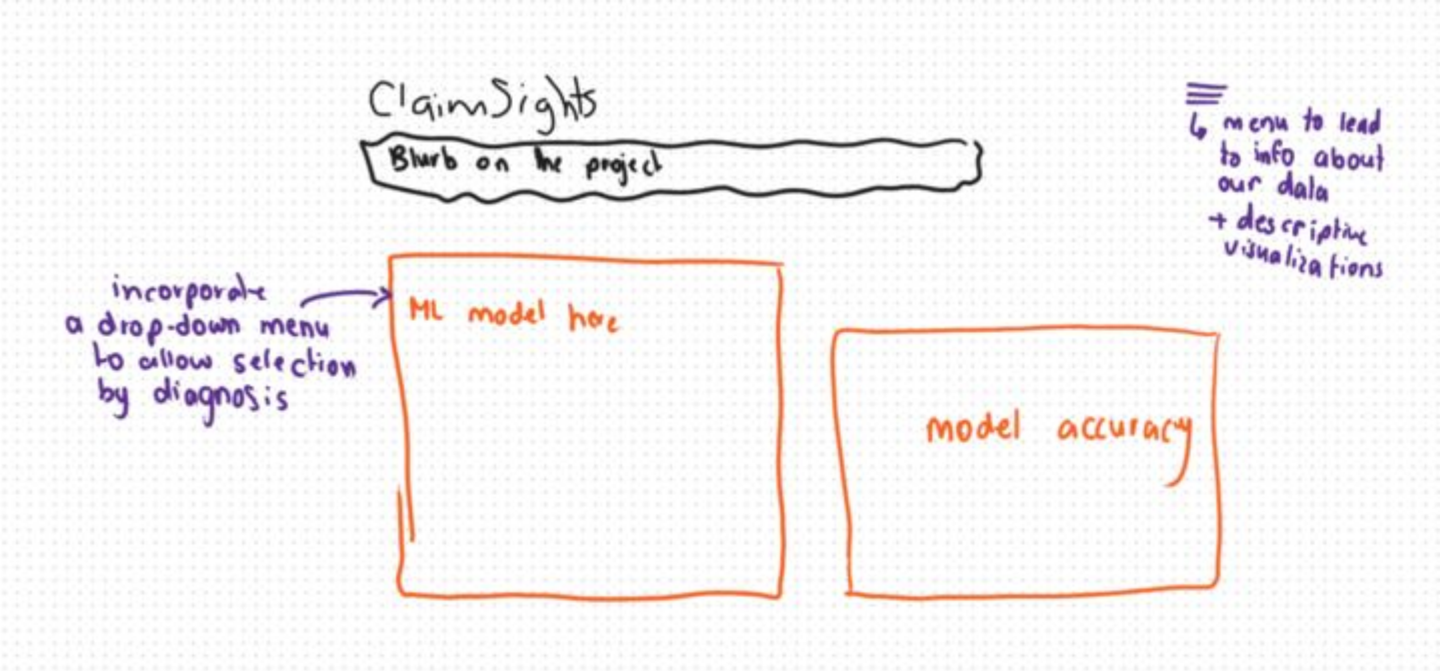
Flask

- Connecting user input to ML model using request, programmed to trigger submission result on HTML

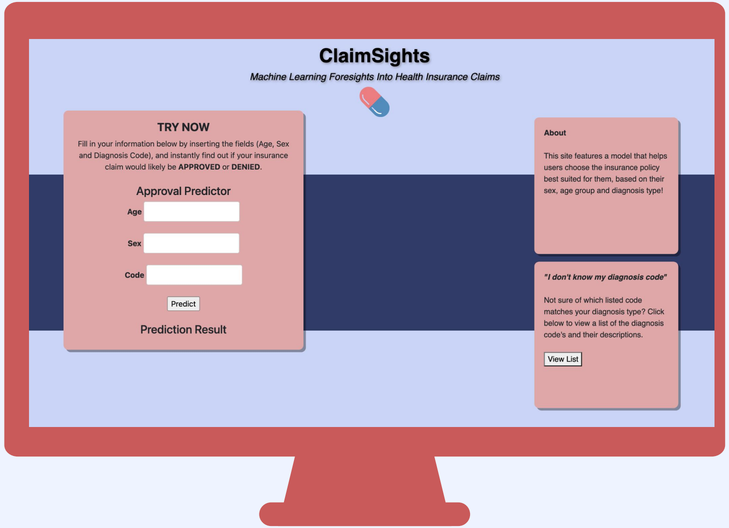
Flask connection to front-end

```
@app.route('/predict', methods=['POST'])
def predict_placement():
        #Reqeusting forms
        age = str(request.form.get('Age'))
        sex = str(request.form.get('Sex'))
        diag = str(request.form.get('DiagnosisType'))
        #retrieveing sex binary code 0 is male and 1 is female
        sex_value = None
        if (sex == 'M'):
            sex_value = 0
        elif (sex == 'F'):
            sex_value = 1
```

ClaimSights Mindmap



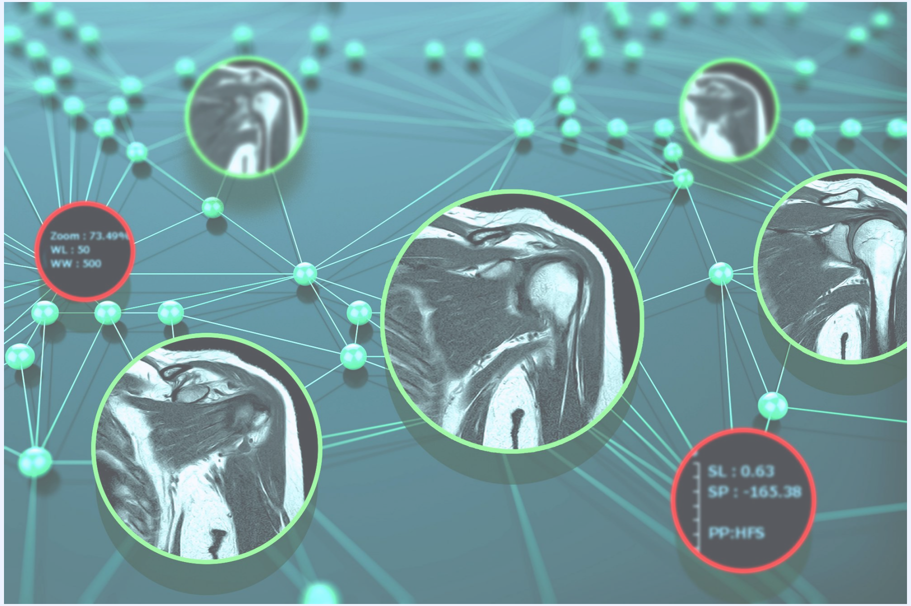
5.2: Live Demonstration Of ClaimSights



6.0

Limitations and Next Steps

6.1: Challenges



- More domain knowledge on diagnoses, diseases and medications needed.
- Overcoming class imbalance - balancing accuracy and recall scores.
- Public live web deployment.

6.2: Nextsteps

Model is specific to Insurance Company X, similar models can be built to other companies and policies.

Thank you

Questions?