Breast Cancer Jump Scare Team 5

Cadeem Musgrove, Lucas Perez,
Jaskirat Singh & Aditya Singh

Agenda

This study aims to create models to determine overall survivability and predict clinical data for clinicians

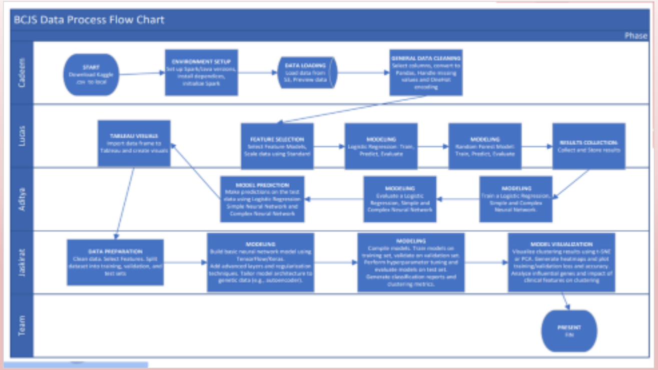
About the Data	Data sources, time range, size, and end to end data flow from source to consumption with components and connectors describing how data evolves through the flow to generate end visuals and for modelling
Exploratory Analysis 1	Modeling Experiment, Evaluation
Exploratory Analysis 2	Modeling Experiment, Evaluation
Exploratory Analysis 3	Modeling Experiment, Evaluation
Considerations & Key Learning	Discussing our key learnings, future implementations, and our key learnings

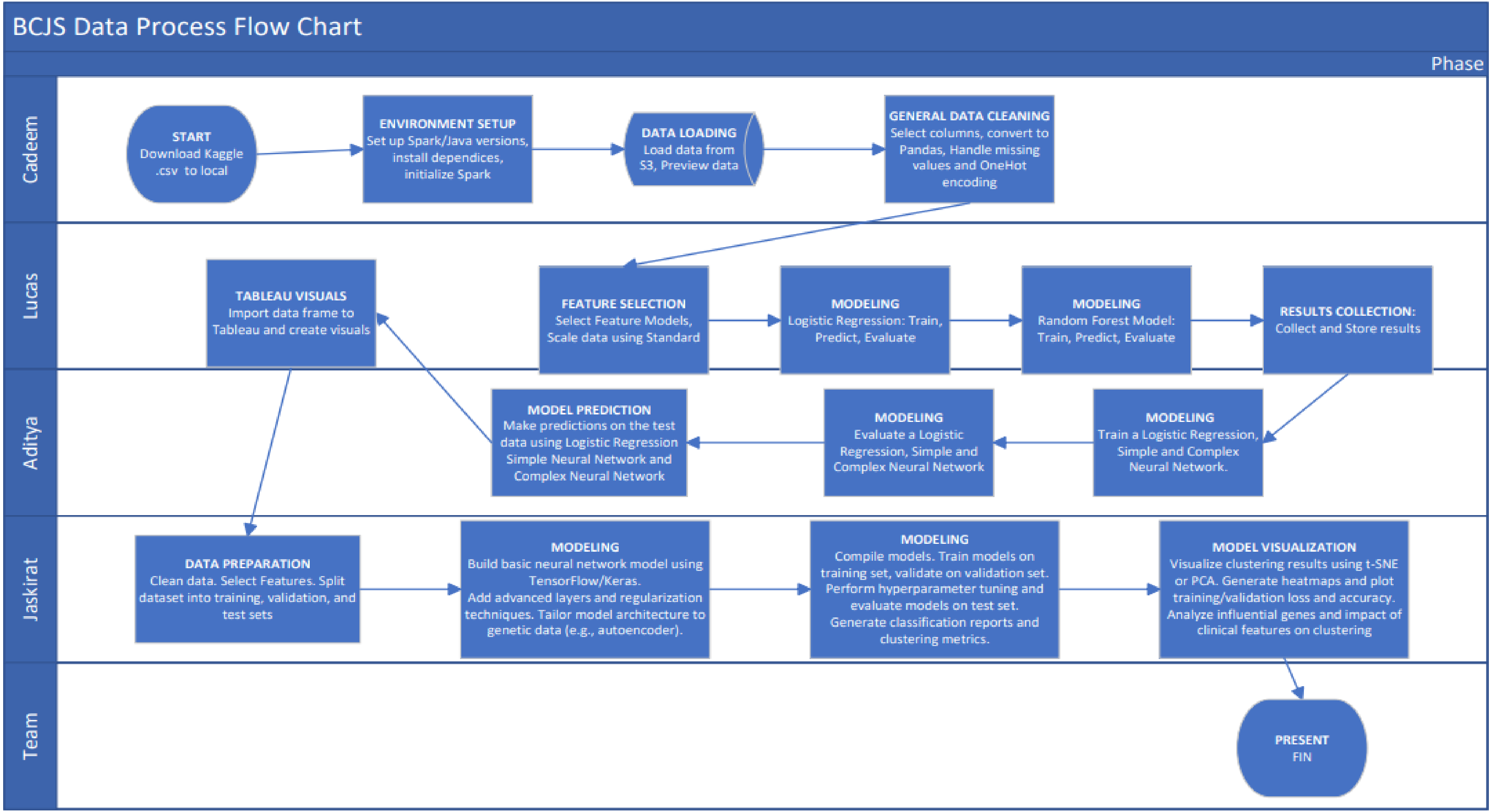
About the Data

Data Description

Sourced from Kaggle. Data is 1904 Rows x 676 Columns. Data is taken between 2016 to 2020

Data Process




BCJS Data Process Flow Chart
Phase
Cadeem
Lucas
Aditya
Jaskirat
Team
START
Download Kaggle .csv to local
ENVIRONMENT SETUP
Set up Spark/Java versions, install dependices, initialize Spark
DATA LOADING
Load data from S3, Preview data
GENERAL DATA CLEANING
Select columns, convert to Pandas, Handle missing values and OneHot encoding
FEATURE SELECTION
Select Feature Models, Scale data using Standard
MODELING
Logistic Regression: Train, Predict, Evaluate
MODELING
Random Forest Model: Train, Predict, Evaluate
RESULTS COLLECTION:
Collect and Store results
MODELING
Train a Logistic Regression, Simple and Complex Neural Network.
MODELING
Evaluate a Logistic Regression, Simple and Complex Neural Network
MODEL PREDICTION
Make predictions on the test data using Logistic Regression Simple Neural Network and Complex Neural Network
TABLEAU VISUALS
Import data frame to Tableau and create visuals
DATA PREPARATION
Clean data. Select Features. Split dataset into training, validation, and test sets
MODELING
Build basic neural network model using TensorFlow/Keras.
Add advanced layers and regularization techniques. Tailor model architecture to genetic data (e.g., autoencoder).
MODELING
Compile models. Train models on training set, validate on validation set. Perform hyperparameter tuning and evaluate models on test set.
Generate classification reports and clustering metrics.
MODEL VISUALIZATION
Visualize clustering results using t-SNE or PCA. Generate heatmaps and plot training/validation loss and accuracy. Analyze influential genes and impact of clinical features on clustering
PRESENT
FIN

Exploratory Analysis 1

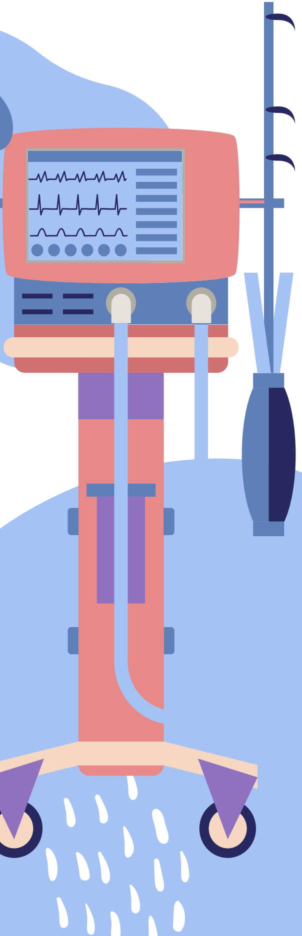
Key Insights on Target Variables

Target variable ‘overall_survival’ is binary. Indicates whether a patient survived over a period of time.

Modeling Experiment Design

Baseline model: Logistic Regression Model.

Random Survival Forest (RSF) Model employed to handle survivability over time (time-to-event data)



Exploratory Analysis 1

Model Evaluation

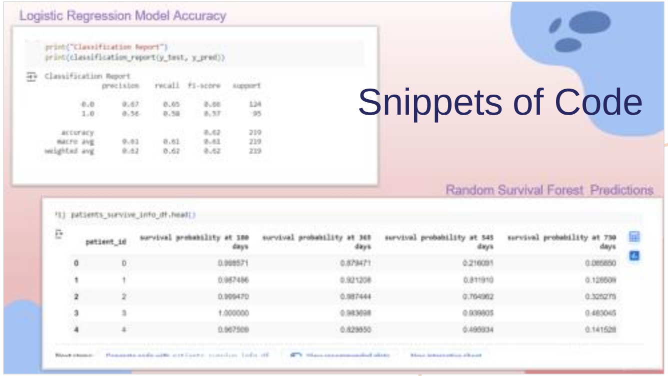
Metrics Used

- Accuracy: Primary metric in Logistic Regression
- Survival Probability Metric: RSF Model different time points (ex. 365 days)

Model Comparison

The RSF model of survival data shows better generalization and performance compared to the baseline logistic regression model.

Snippets of Code


Logistic Regression Model Accuracy

```
print("Classification Report")
print(classification_report(y_test, y_pred))
```

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.67      0.65      0.66       124
         1.0       0.56      0.58      0.57        95

    accuracy                           0.62       219
   macro avg       0.61      0.61      0.61       219
weighted avg       0.62      0.62      0.62       219
```

Random Survival Forest Predictions

```
patients_survive_info_df.head()
```

	patient_id	survival probability at 180 days	survival probability at 365 days	survival probability at 545 days	survival probability at 730 days
0	0	0.998571	0.879471	0.216091	0.085850
1	1	0.987486	0.921208	0.811910	0.128509
2	2	0.999470	0.987444	0.764962	0.325275
3	3	1.000000	0.983698	0.939805	0.483045
4	4	0.967509	0.829850	0.495934	0.141528

Logistic Regression Model Accuracy

```
print("Classification Report")
print(classification_report(y_test, y_pred))
```

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.67      0.65      0.66       124
         1.0       0.56      0.58      0.57        95

    accuracy                           0.62       219
   macro avg       0.61      0.61      0.61       219
weighted avg       0.62      0.62      0.62       219
```

Random Survival Forest Predictions

```
71] patients_survive_info_df.head()
```

	patient_id	survival probability at 180 days	survival probability at 365 days	survival probability at 545 days	survival probability at 730 days
0	0	0.998571	0.879471	0.216091	0.085850
1	1	0.987496	0.921208	0.811910	0.128509
2	2	0.999470	0.987444	0.764962	0.325275
3	3	1.000000	0.983698	0.939805	0.483045
4	4	0.967509	0.829850	0.495934	0.141528

Next steps: Generate code with patients_survive_info_df | View recommended plots | New interactive sheet

Exploratory Analysis 2

Key Insights

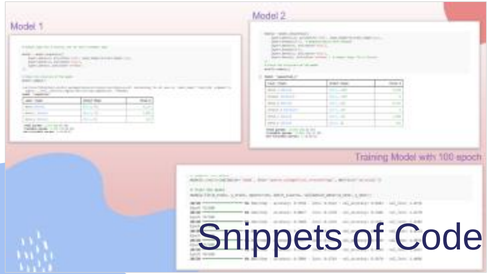
Type of cancer and type of treatment, which treatment is more effective against certain types of cancers.

Model Design

- Created Dummys for testing predictions.
- Ran Logistics Regression as the measurement is categorical

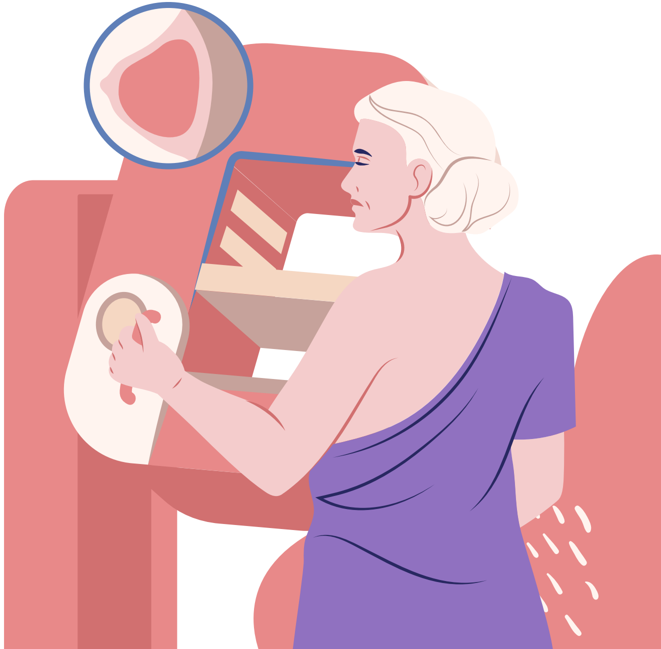
Model evaluation

- Ran with different epochs, neurons, and layers.
- To achieve 56% accuracy.



Snippets of Code

Tableau Visuals



Model 1

```python
# Output layer has 8 neurons, one for each treatment type

model1 = models.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(8, activation='softmax')
])

# Check the structure of the model
model1.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	4,224
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 8)	264

```
Total params: 6,568 (25.66 KB)
Trainable params: 6,568 (25.66 KB)
Non-trainable params: 0 (0.00 B)
```

Model 2

```python
model2 = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dropout(0.5),  # Regularization with Dropout
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(32, activation='relu'),
    layers.Dense(8, activation='softmax')  # Output layer for 8 classes
])
# Check the structure of the model
model2.summary()
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 128)	8,448
dropout (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 32)	2,080
dense_6 (Dense)	(None, 8)	264

```
Total params: 19,048 (74.41 KB)
Trainable params: 19,048 (74.41 KB)
Non-trainable params: 0 (0.00 B)
```

Training Model with 100 epoch

```python
# Compile the model
model2.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model2.fit(X_train, y_train, epochs=100, batch_size=64, validation_data=(X_test, y_test))
```

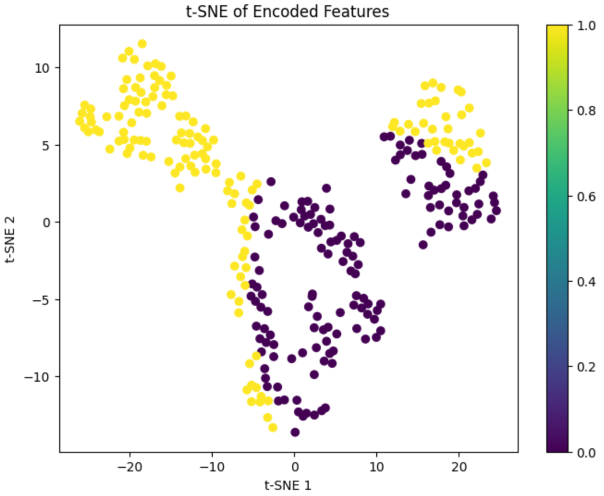
```
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7930 - loss: 0.5112 - val_accuracy: 0.5643 - val_loss: 1.4376
Epoch 73/100
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8047 - loss: 0.5278 - val_accuracy: 0.5486 - val_loss: 1.4270
Epoch 74/100
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7848 - loss: 0.5254 - val_accuracy: 0.5643 - val_loss: 1.4384
Epoch 75/100
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7992 - loss: 0.4972 - val_accuracy: 0.5580 - val_loss: 1.4372
Epoch 76/100
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7997 - loss: 0.5350 - val_accuracy: 0.5799 - val_loss: 1.4075
Epoch 77/100
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7959 - loss: 0.5370 - val_accuracy: 0.5549 - val_loss: 1.4267
Epoch 78/100
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.7809 - loss: 0.5714 - val_accuracy: 0.5674 - val_loss: 1.4606
```

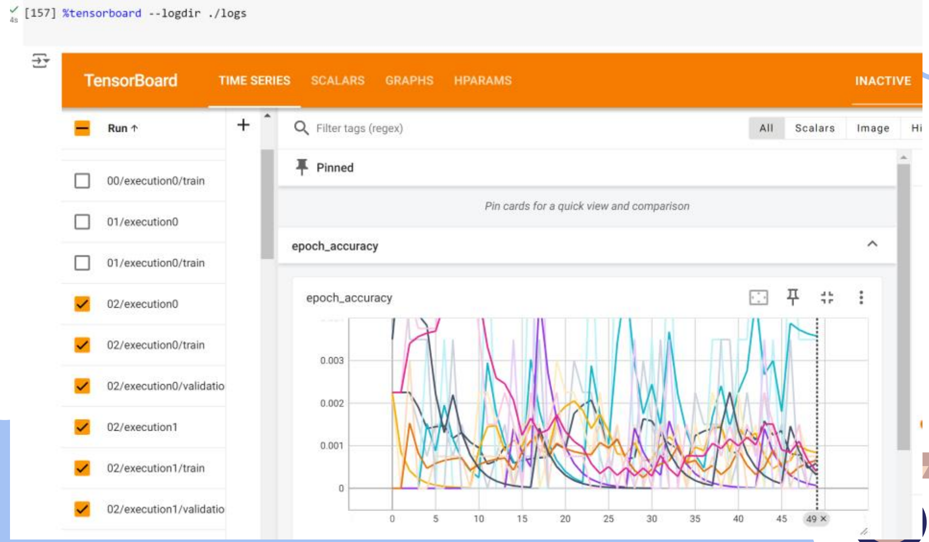
Exploratory Analysis 3

Key Insights

Metrics Used: Loss and Accuracy for models and clustering metrics (silhouette score and Davies Bouldin score

Use of TensorBoard for following Model training.





Considerations & Key Lessons

Time

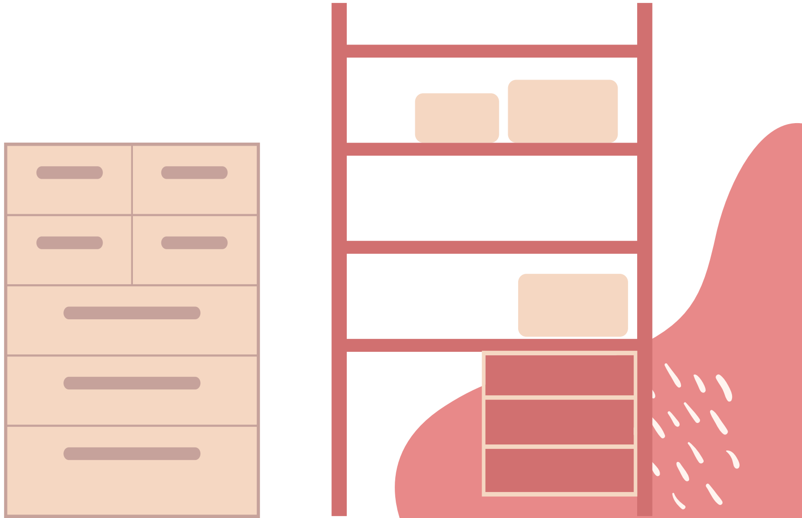
Horizon of the study
Time constrain on the project time line

Scope

11 Models in 2 weeks

Domain Knowledge

Domain expertise in the medical industry. This model can help clinicians deal with better Prognosis for patiensts



Questions?