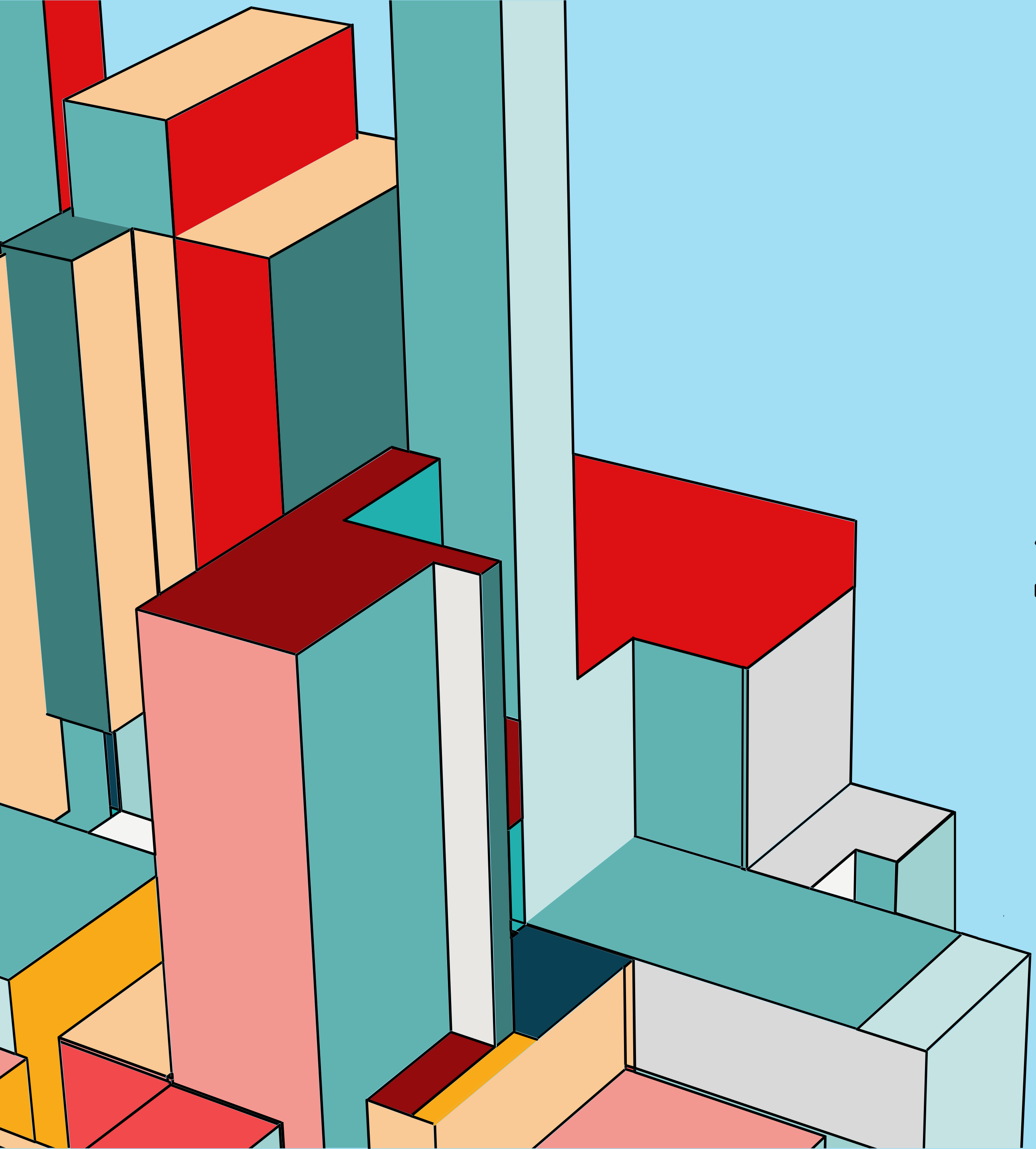


# RARE DISEASE DETECTION FROM STANDARD LAB TESTING

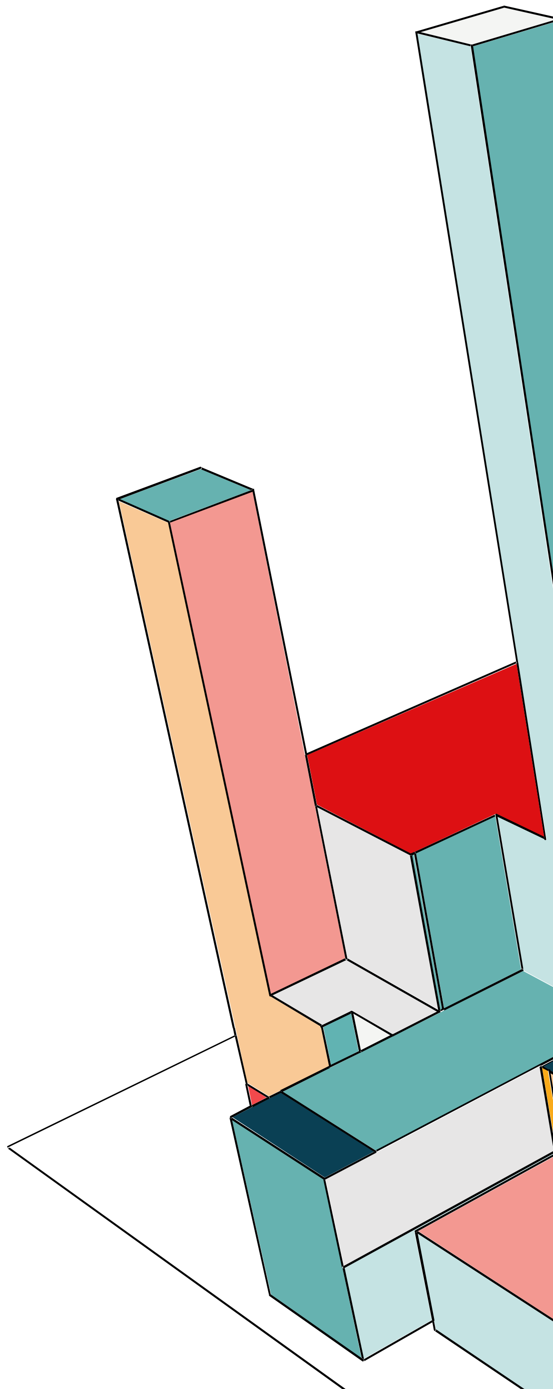
Maina Musa

University of Texas at Austin

April 2025

## SCHEMA

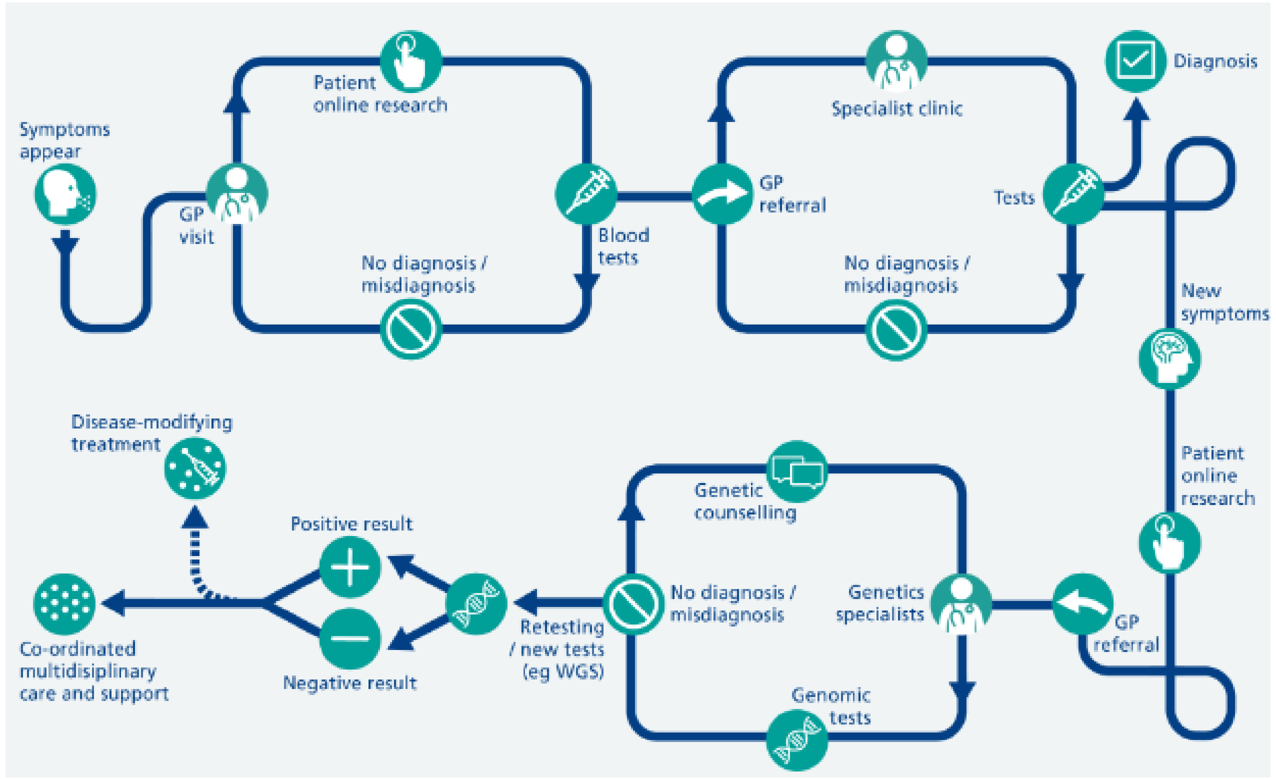
- Introduction
- Methodology
- Results
- Demo
- Future Directions

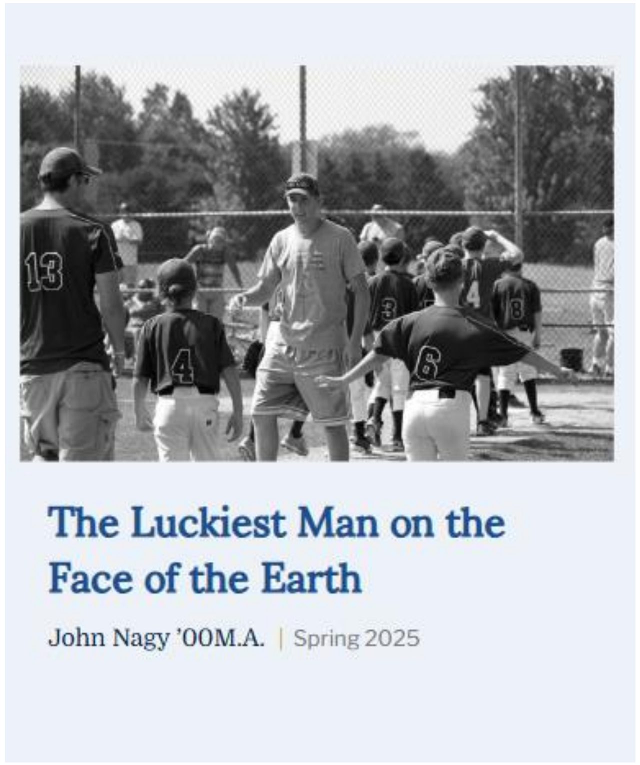


## INTRODUCTION: FIGHTING FOR THOSE WITH RARE DISEASES

WHAT WOULD
YOU FIGHT FOR?
Fighting
for those with
rare diseases
Rare disease research can be as uncommon as the diseases themselves, but Notre Dame is committed to understanding, treating, and advocating for those affected by rare diseases
UNIVERSITY OF NOTRE DAME

# INTRODUCTION: CLOSER TO HOME

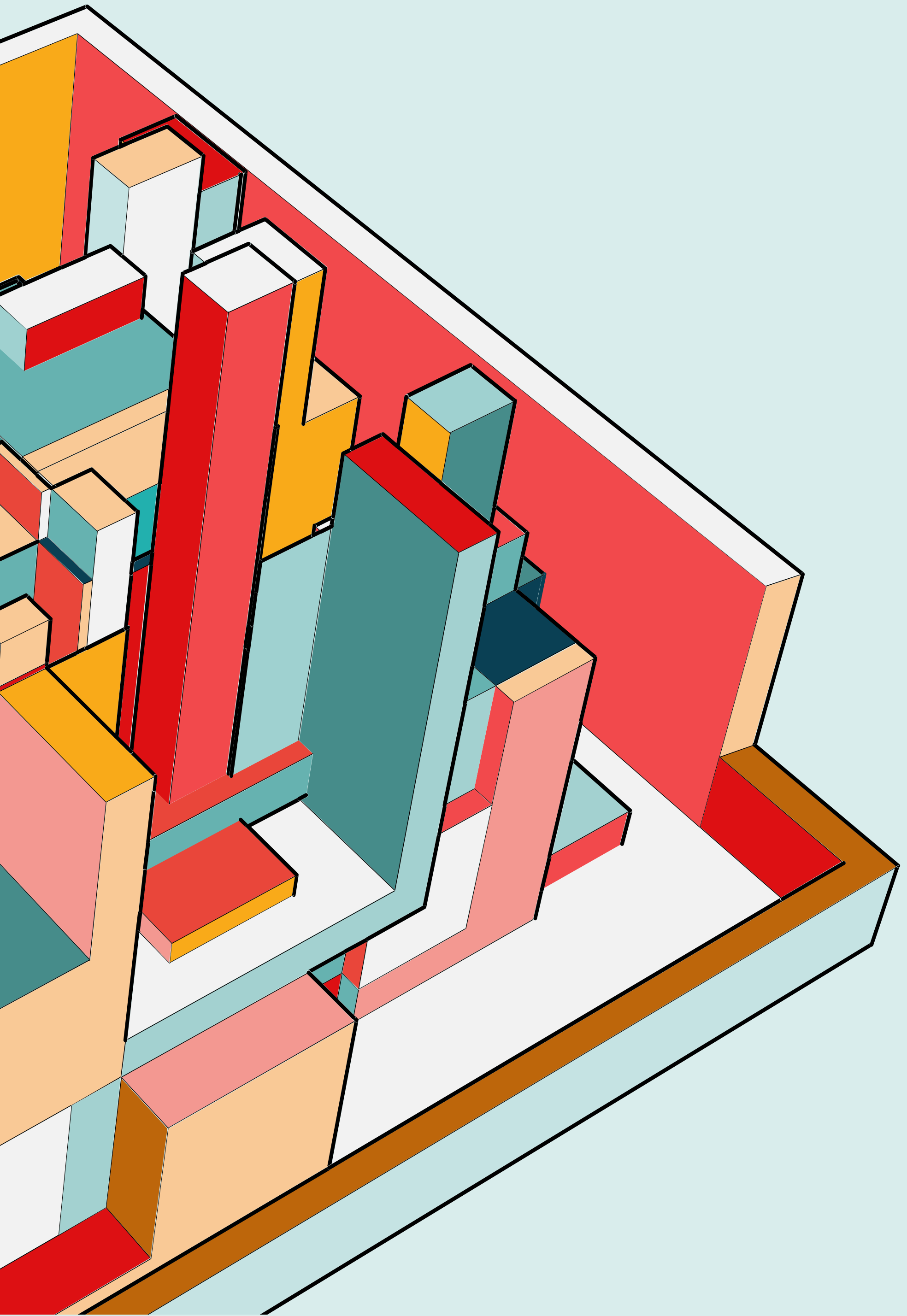




The Luckiest Man on the Face of the Earth

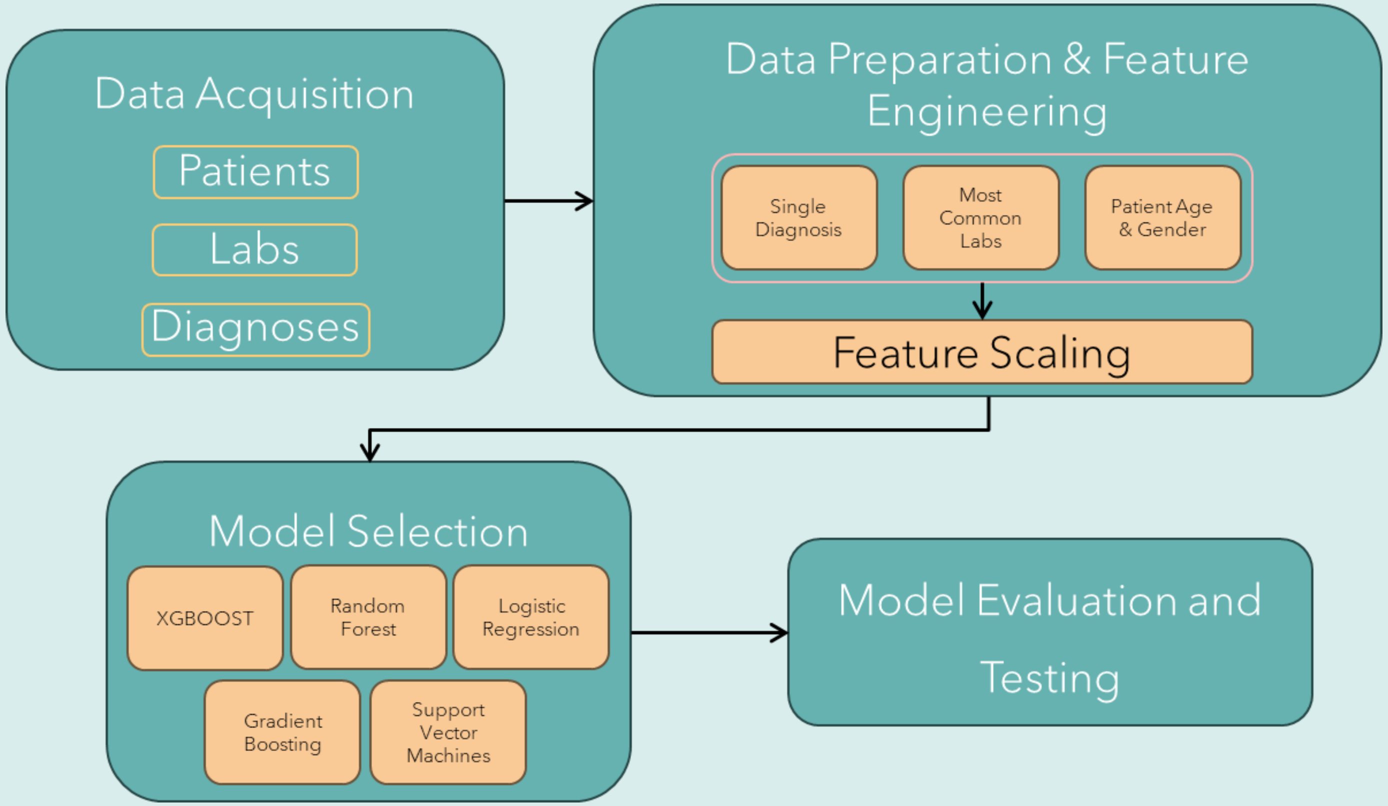
John Nagy '00M.A. | Spring 2025

[Diagnostic odyssey in rare disease – Knowledge Hub]

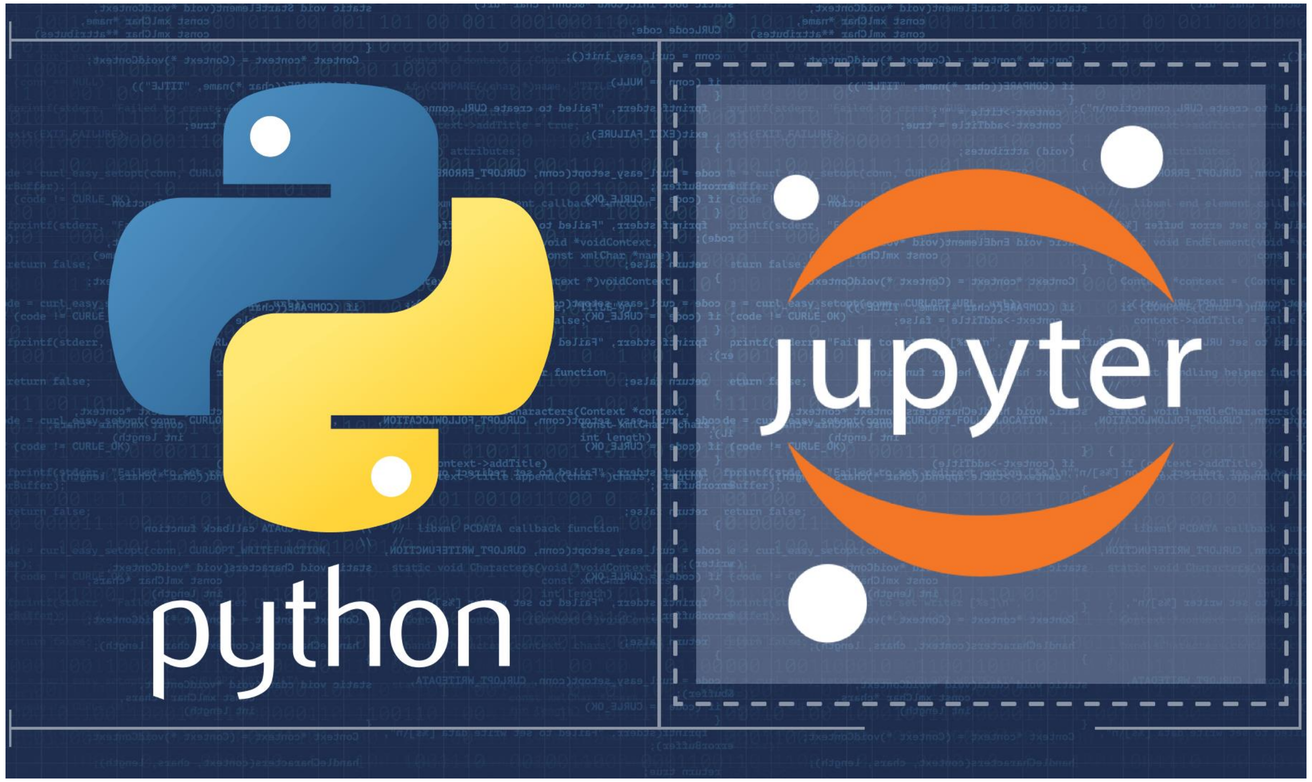


## METHODOLOGY

Steps to procure data, set up experiments, and evaluate results



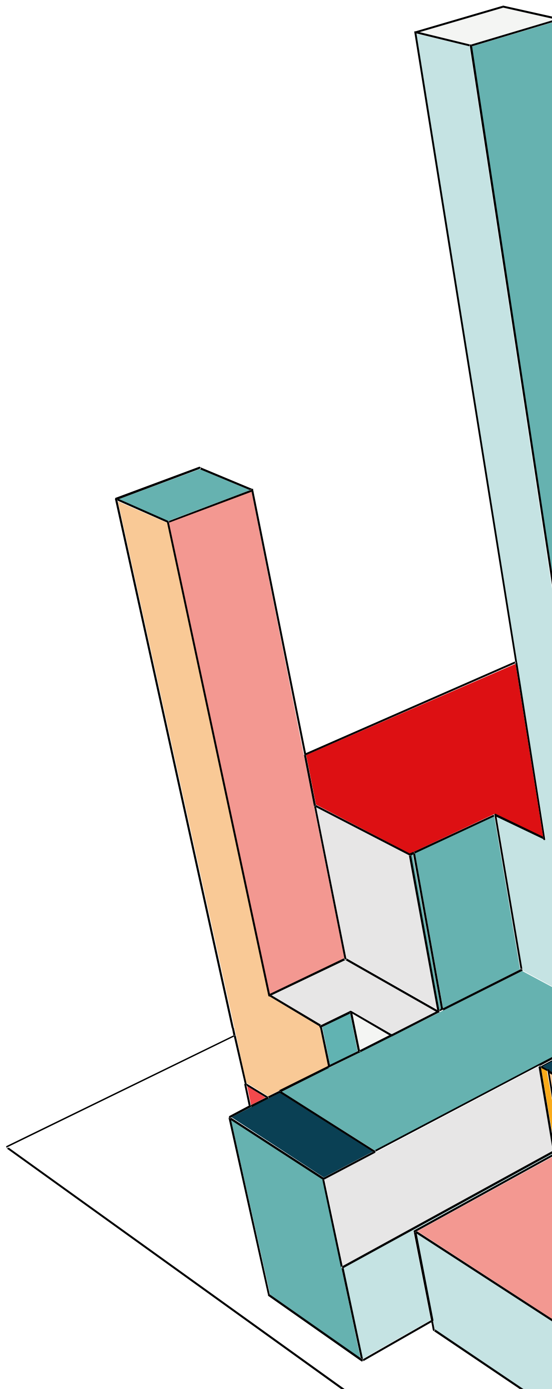
## METHODOLOGY: DATA ACQUISITION



- Used the MIMIC-III Clinical Database, provided by PhysioNet as the model's data source
- Comprehensive retrieval of tables including patients, lab events, lab items, diagnosis ICD, and diagnosis items
- Local copies of the gzipped CSV files were downloaded and imported into a Python notebook via the pandas package

## METHODOLOGY: DATA PREPARATION & FEATURE ENGINEERING

- Due to the unavailability of resources listing ICD-9 codes for diseases considered rare (less than 200,000 diagnoses), rare diseases were defined as those appearing only once in the dataset
- A threshold was set to identify the most common labs, defined as the top 10 most frequent labs
- Gender data was extracted from the patients table and mapped to a binary column (1 or 0)
- Age was calculated by subtracting the chart time from the lab admission date from the patient's date of birth
- The MinMaxScaler from scikit-learn was utilized to standardize the data to a coherent scale



## METHODOLOGY: DATA IMBALANCE & MODEL SELECTION

- First used SMOTE then ADASYN to help deal with target variable imbalances
- Used grid search on multiple algorithm types

| Algorithm | Result |
|---|---|
| RandomForestClassifier | Poor |
| LogisticRegression | Very Poor |
| SVC | Very Poor |
| GradientBoostingClassifier | Very Poor |
| XGBClassifier | Crash Machine |

## RESULTS

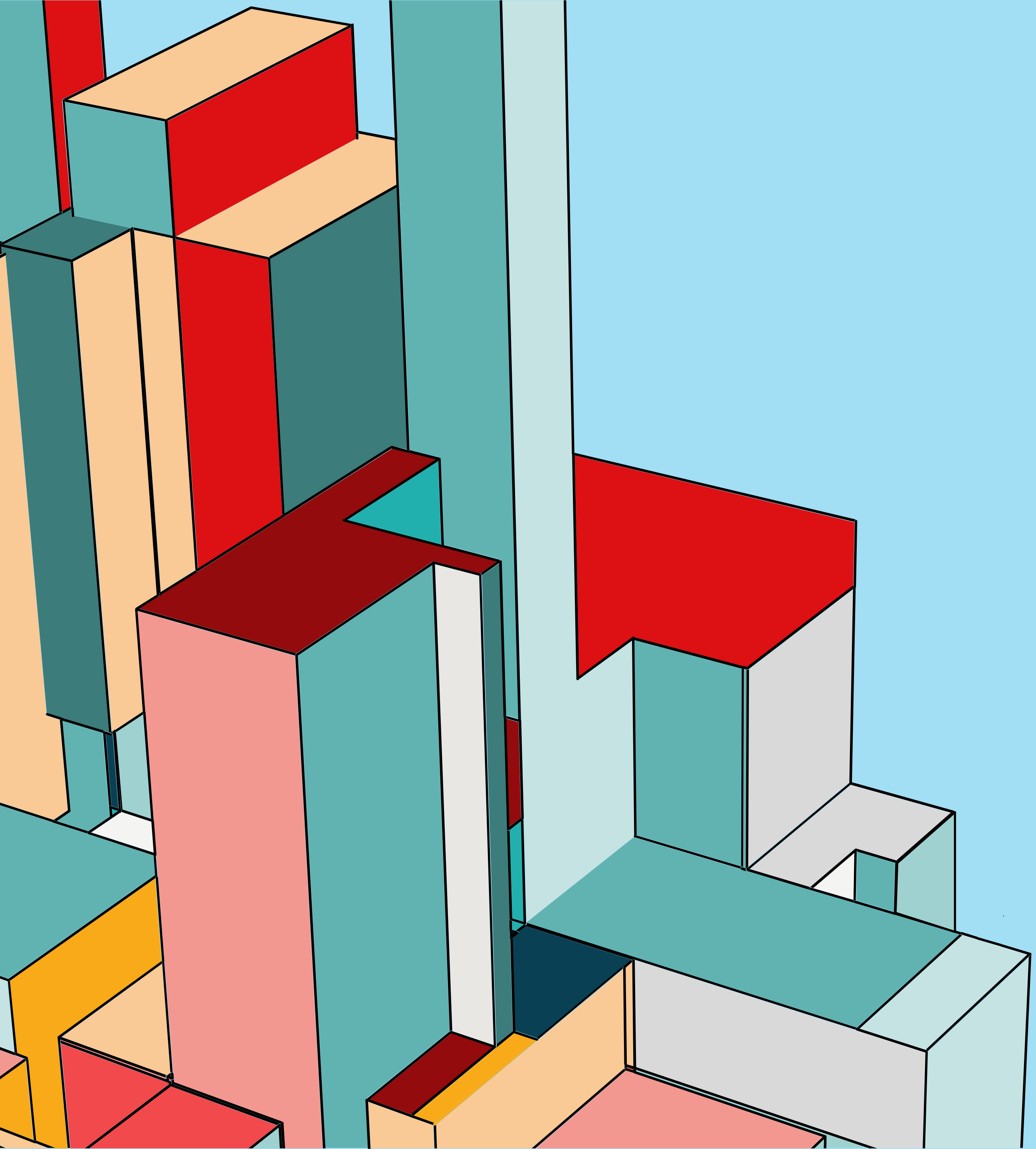
## BEST RESULTS: RANDOM FOREST

Classification Report:

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0.0 | 0.98 | 0.88 | 0.93 | 11523 |
| 1.0 | 0.05 | 0.27 | 0.09 | 273 |
| | | | | |
| Accuracy | | | 0.87 | 11796 |
| Macro Avg | 0.52 | 0.58 | 0.51 | 11796 |
| Weighted Avg | 0.96 | 0.87 | 0.91 | 11796 |
| | | | | |
| Accuracy Score: | | | | |
| 0.869616819261 | | | | |

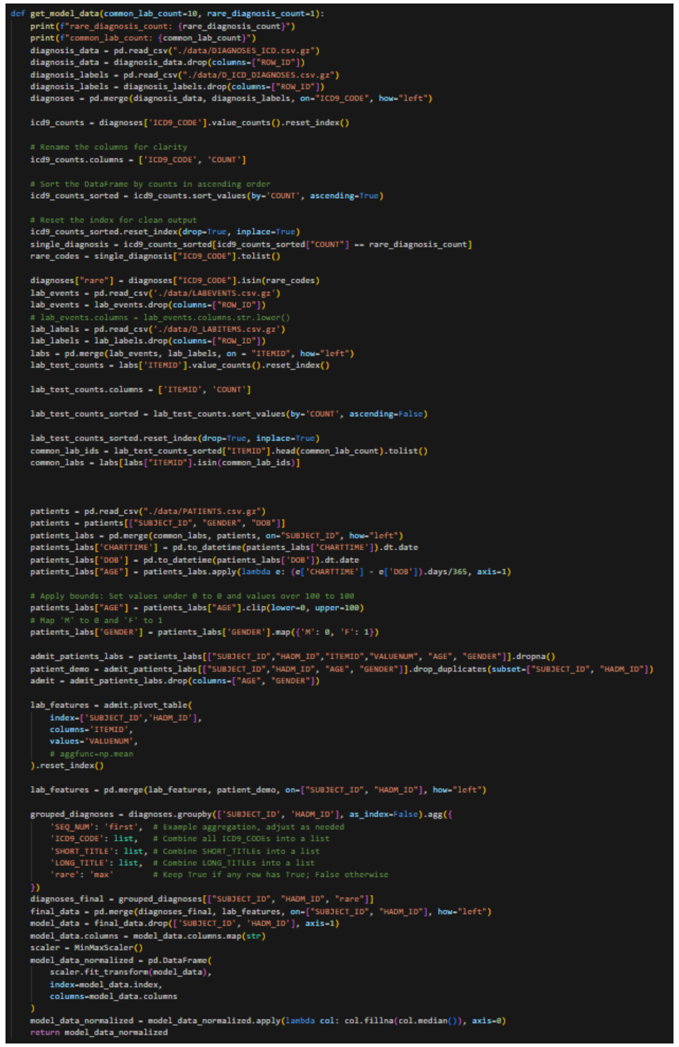
Best results were obtained with a Random Forest Classifier with the hyperparams:

- 250 estimators
- 14 max depth
- 5 minimum samples split

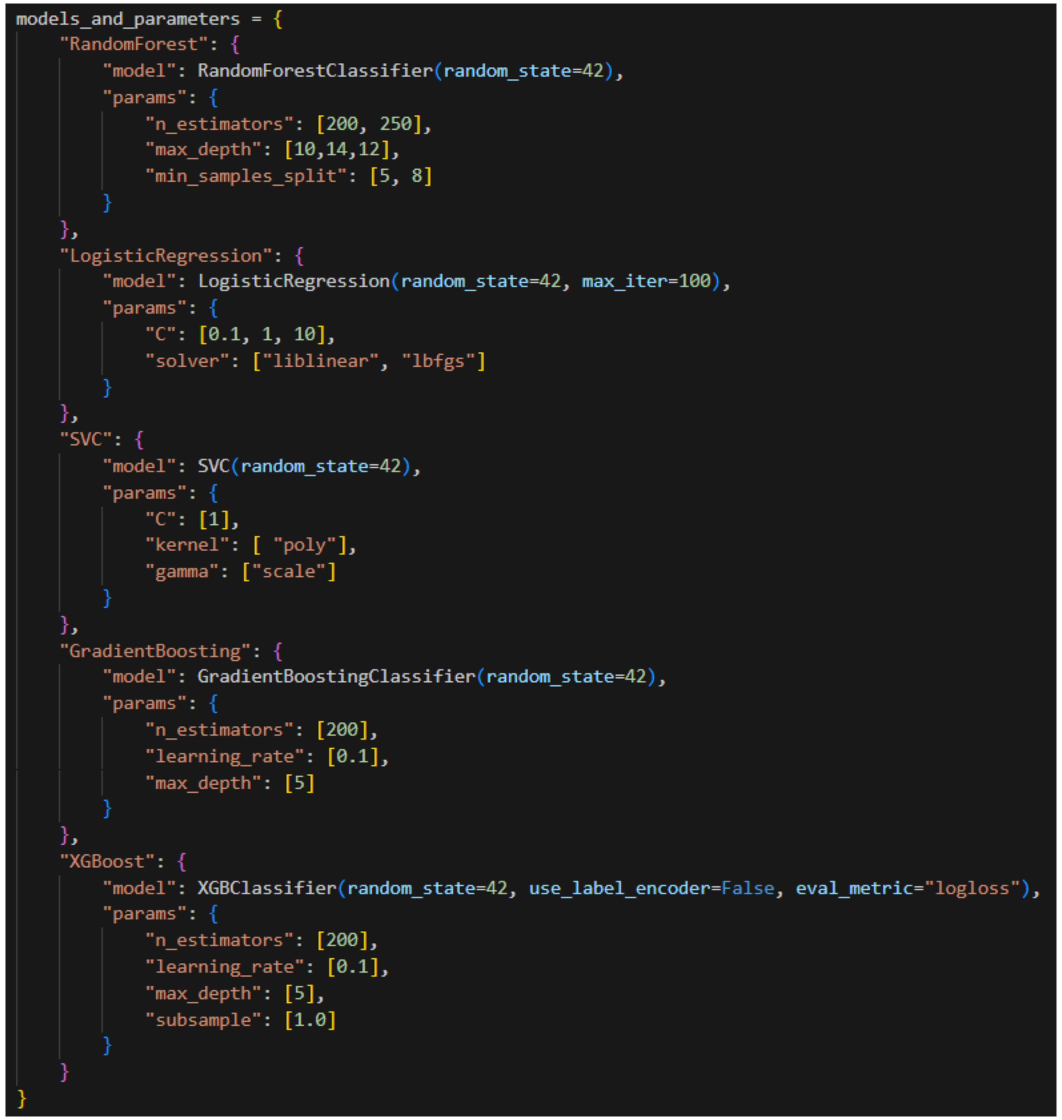


## DEMO

# DEMO: MODEL DATA & ALGORITHMS

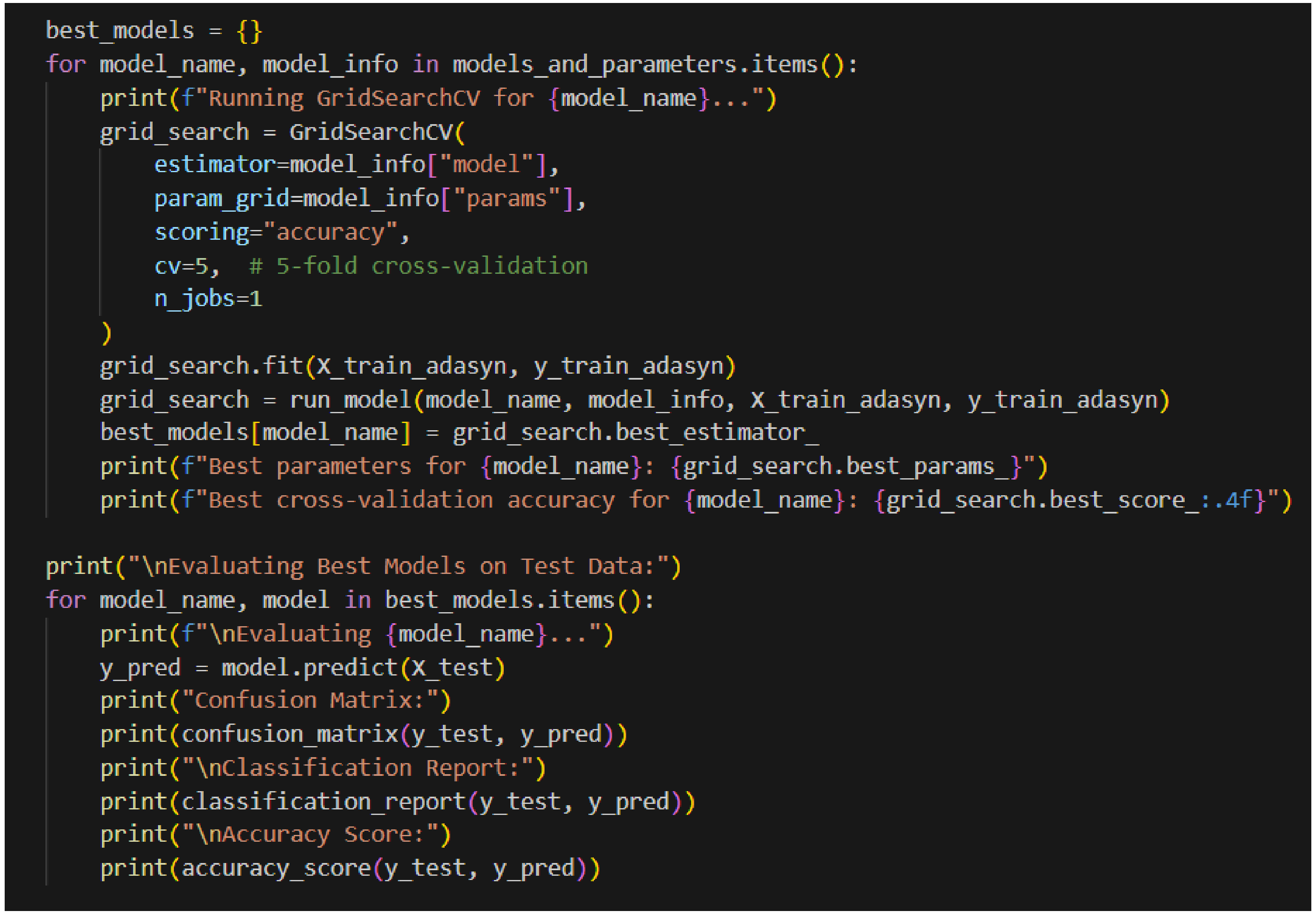


```python
def get_model_data(common_lab_count=10, rare_diagnosis_count=1):
    print(f"rare_diagnosis_count: {rare_diagnosis_count}")
    print(f"common_lab_count: {common_lab_count}")
    diagnosis_data = pd.read_csv("./data/DIAGNOSES_ICD.csv.gz")
    diagnosis_data = diagnosis_data.drop(columns=["ROW_ID"])
    diagnosis_labels = pd.read_csv("./data/D_ICD_DIAGNOSES.csv.gz")
    diagnosis_labels = diagnosis_labels.drop(columns=["ROW_ID"])
    diagnoses = pd.merge(diagnosis_data, diagnosis_labels, on="ICD9_CODE", how="left")

    icd9_counts = diagnoses['ICD9_CODE'].value_counts().reset_index()

    # Rename the columns for clarity
    icd9_counts.columns = ['ICD9_CODE', 'COUNT']

    # Sort the DataFrame by counts in ascending order
    icd9_counts_sorted = icd9_counts.sort_values(by='COUNT', ascending=True)

    # Reset the index for clean output
    icd9_counts_sorted.reset_index(drop=True, inplace=True)
    single_diagnosis = icd9_counts_sorted[icd9_counts_sorted["COUNT"] == rare_diagnosis_count]
    rare_codes = single_diagnosis["ICD9_CODE"].tolist()

    diagnoses["rare"] = diagnoses["ICD9_CODE"].isin(rare_codes)
    lab_events = pd.read_csv('./data/LABEVENTS.csv.gz')
    lab_events = lab_events.drop(columns=["ROW_ID"])
    # lab_events.columns = lab_events.columns.str.lower()
    lab_labels = pd.read_csv('./data/D_LABITEMS.csv.gz')
    lab_labels = lab_labels.drop(columns=["ROW_ID"])
    labs = pd.merge(lab_events, lab_labels, on = "ITEMID", how="left")
    lab_test_counts = labs['ITEMID'].value_counts().reset_index()

    lab_test_counts.columns = ['ITEMID', 'COUNT']

    lab_test_counts_sorted = lab_test_counts.sort_values(by='COUNT', ascending=False)

    lab_test_counts_sorted.reset_index(drop=True, inplace=True)
    common_lab_ids = lab_test_counts_sorted["ITEMID"].head(common_lab_count).tolist()
    common_labs = labs[labs["ITEMID"].isin(common_lab_ids)]



    patients = pd.read_csv("./data/PATIENTS.csv.gz")
    patients = patients[["SUBJECT_ID", "GENDER", "DOB"]]
    patients_labs = pd.merge(common_labs, patients, on="SUBJECT_ID", how="left")
    patients_labs['CHARTTIME'] = pd.to_datetime(patients_labs['CHARTTIME']).dt.date
    patients_labs['DOB'] = pd.to_datetime(patients_labs['DOB']).dt.date
    patients_labs["AGE"] = patients_labs.apply(lambda e: (e['CHARTTIME'] - e['DOB']).days/365, axis=1)

    # Apply bounds: Set values under 0 to 0 and values over 100 to 100
    patients_labs["AGE"] = patients_labs["AGE"].clip(lower=0, upper=100)
    # Map 'M' to 0 and 'F' to 1
    patients_labs['GENDER'] = patients_labs['GENDER'].map({'M': 0, 'F': 1})

    admit_patients_labs = patients_labs[["SUBJECT_ID","HADM_ID","ITEMID","VALUENUM", "AGE", "GENDER"]].dropna()
    patient_demo = admit_patients_labs[["SUBJECT_ID","HADM_ID", "AGE", "GENDER"]].drop_duplicates(subset=["SUBJECT_ID", "HADM_ID"])
    admit = admit_patients_labs.drop(columns=["AGE", "GENDER"])

    lab_features = admit.pivot_table(
        index=['SUBJECT_ID','HADM_ID'],
        columns='ITEMID',
        values='VALUENUM',
        # aggfunc=np.mean
    ).reset_index()

    lab_features = pd.merge(lab_features, patient_demo, on=["SUBJECT_ID", "HADM_ID"], how="left")

    grouped_diagnoses = diagnoses.groupby(['SUBJECT_ID', 'HADM_ID'], as_index=False).agg({
        'SEQ_NUM': 'first',  # Example aggregation, adjust as needed
        'ICD9_CODE': list,   # Combine all ICD9_CODEs into a list
        'SHORT_TITLE': list, # Combine SHORT_TITLEs into a list
        'LONG_TITLE': list,  # Combine LONG_TITLEs into a list
        'rare': 'max'        # Keep True if any row has True; False otherwise
    })
    diagnoses_final = grouped_diagnoses[["SUBJECT_ID", "HADM_ID", "rare"]]
    final_data = pd.merge(diagnoses_final, lab_features, on=["SUBJECT_ID", "HADM_ID"], how="left")
    model_data = final_data.drop(['SUBJECT_ID', 'HADM_ID'], axis=1)
    model_data.columns = model_data.columns.map(str)
    scaler = MinMaxScaler()
    model_data_normalized = pd.DataFrame(
        scaler.fit_transform(model_data),
        index=model_data.index,
        columns=model_data.columns
    )
    model_data_normalized = model_data_normalized.apply(lambda col: col.fillna(col.median()), axis=0)
    return model_data_normalized
```

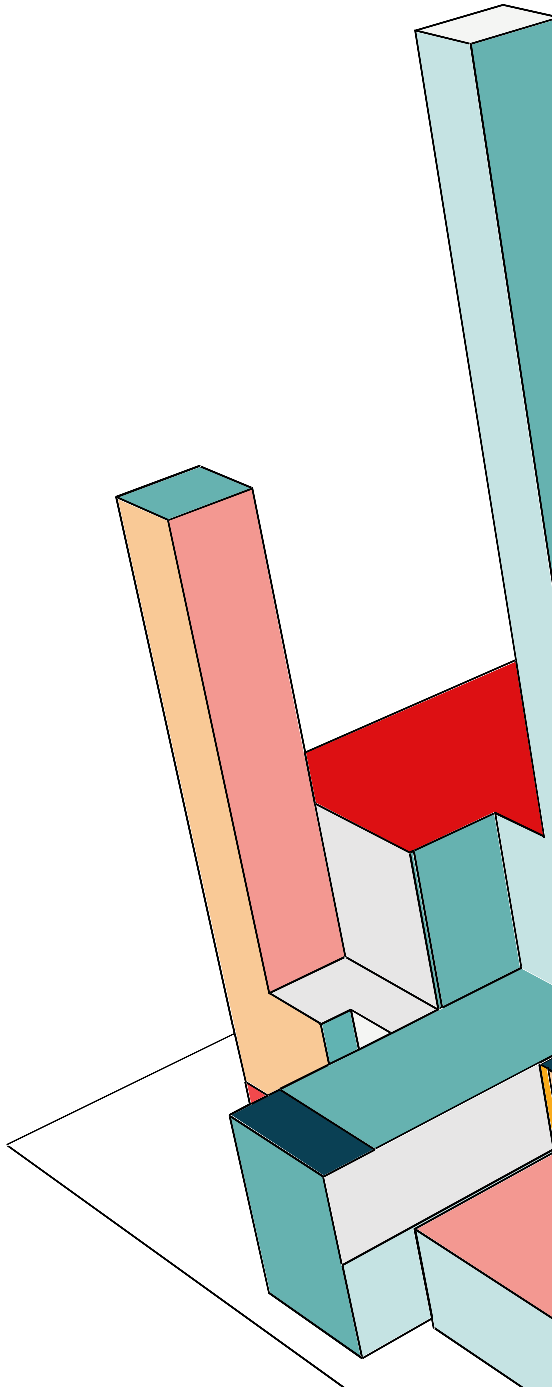


```python
models_and_parameters = {
    "RandomForest": {
        "model": RandomForestClassifier(random_state=42),
        "params": {
            "n_estimators": [200, 250],
            "max_depth": [10,14,12],
            "min_samples_split": [5, 8]
        }
    },
    "LogisticRegression": {
        "model": LogisticRegression(random_state=42, max_iter=100),
        "params": {
            "C": [0.1, 1, 10],
            "solver": ["liblinear", "lbfgs"]
        }
    },
    "SVC": {
        "model": SVC(random_state=42),
        "params": {
            "C": [1],
            "kernel": [ "poly"],
            "gamma": ["scale"]
        }
    },
    "GradientBoosting": {
        "model": GradientBoostingClassifier(random_state=42),
        "params": {
            "n_estimators": [200],
            "learning_rate": [0.1],
            "max_depth": [5]
        }
    },
    "XGBoost": {
        "model": XGBClassifier(random_state=42, use_label_encoder=False, eval_metric="logloss"),
        "params": {
            "n_estimators": [200],
            "learning_rate": [0.1],
            "max_depth": [5],
            "subsample": [1.0]
        }
    }
}
```

## DEMO: MODEL SELECTION



```python
best_models = {}
for model_name, model_info in models_and_parameters.items():
    print(f"Running GridSearchCV for {model_name}...")
    grid_search = GridSearchCV(
        estimator=model_info["model"],
        param_grid=model_info["params"],
        scoring="accuracy",
        cv=5,  # 5-fold cross-validation
        n_jobs=1
    )
    grid_search.fit(X_train_adasyn, y_train_adasyn)
    grid_search = run_model(model_name, model_info, X_train_adasyn, y_train_adasyn)
    best_models[model_name] = grid_search.best_estimator_
    print(f"Best parameters for {model_name}: {grid_search.best_params_}")
    print(f"Best cross-validation accuracy for {model_name}: {grid_search.best_score_:.4f}")

print("\nEvaluating Best Models on Test Data:")
for model_name, model in best_models.items():
    print(f"\nEvaluating {model_name}...")
    y_pred = model.predict(X_test)
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print("\nAccuracy Score:")
    print(accuracy_score(y_test, y_pred))
```

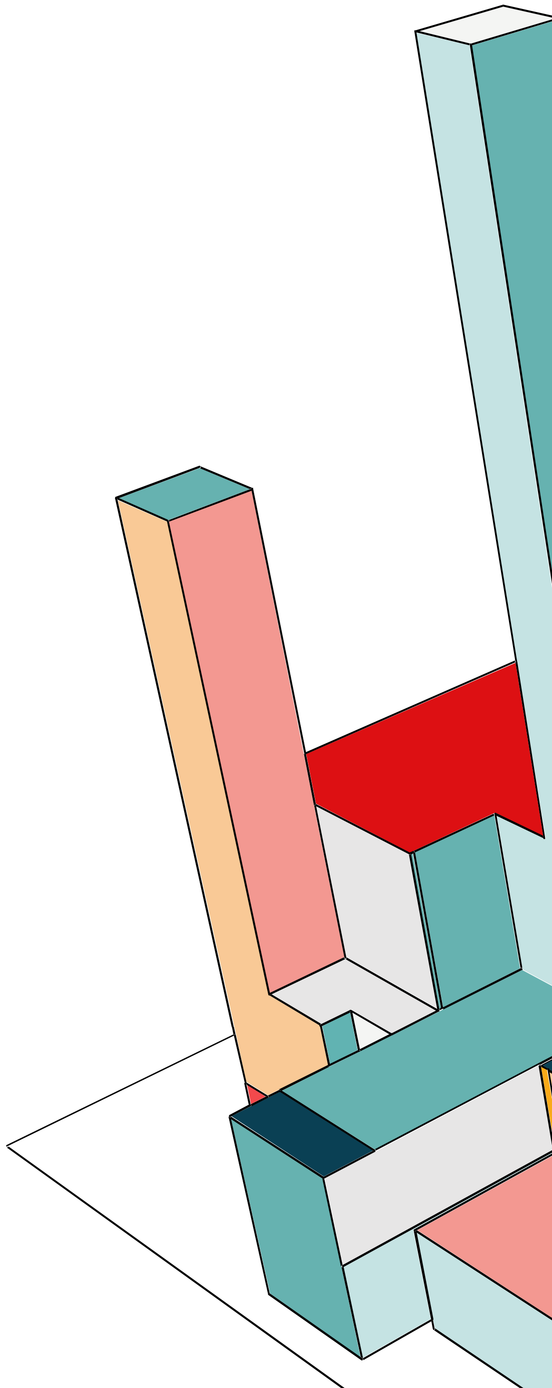


## FUTURE DIRECTIONS

Further ideation

## FUTURE DIRECTIONS

- Exploring different thresholds for defining 'most common' tests beyond the top 10.
- Identifying the true list of ICD-9 codes for rare diseases.
- Incorporating additional demographic information to enrich the dataset.
- Including common chart data.
- Utilizing more powerful computational resources for modeling.
- Applying deep learning techniques to the data.
- Conducting a random search followed by a grid search to refine hyperparameters.

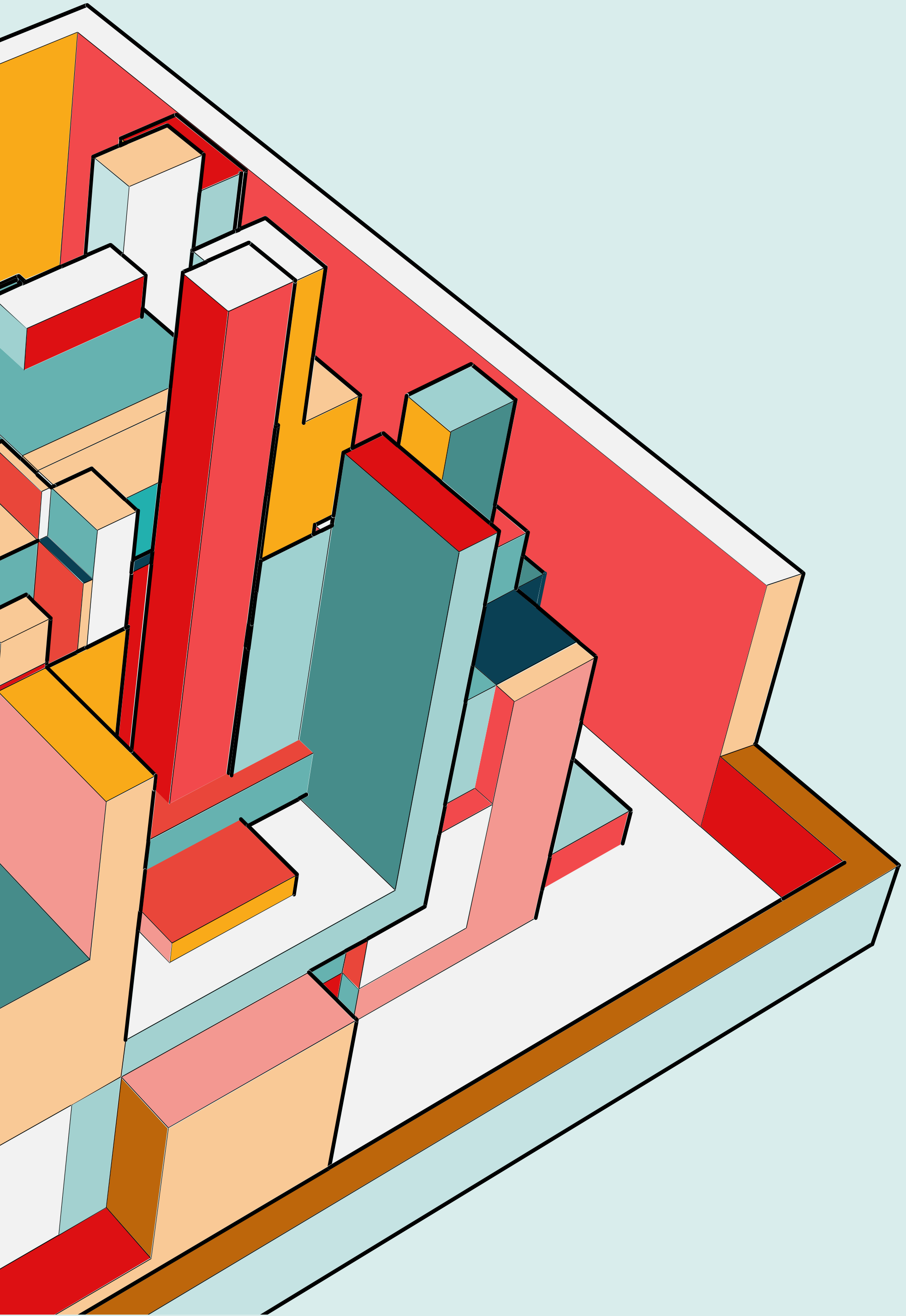


# THANK YOU

Maina Musa

University of Texas at Austin

April 2025



## APPENDIX

References
Flowcharts
Diagrams

## REFERENCES

[1] RareDisease.net. "Getting a Rare Disease Diagnosis." Available: https://raredisease.net/diagnosis.

[2] J. Benito-Lozano, G. Arias-Merino, M. Gómez-Martínez, A. Ancochea-Díaz, A. Aparicio-García, M. Posada de la Paz, and V. Alonso-Ferreira. "Diagnostic Process in Rare Diseases: Determinants Associated with Diagnostic Delay." Int. J. Environ. Res. Public Health, vol. 19, no. 11, p. 6456, 2022. Available: https://www.mdpi.com/1660-4601/19/11/6456.

[3] ScienceDirect. "Search for low-mass resonances decaying into two jets and produced in association with a photon using pp collisions at s=13 TeV with the ATLAS detector." Phys. Lett. B, vol. 795, pp. 56-75, 2019. Available: https://www.sciencedirect.com/science/article/pii/S0012369218300643.

[4] R. Giugliani, S. Castillo Taucher, S. Hafez, J. B. Oliveira, M. Rico-Restrepo, P. Rozenfeld, I. Zarante, and C. Gonzaga-Jauregui. "Opportunities and challenges for newborn screening and early diagnosis of rare diseases in Latin America." Front. Genet., vol. 13, 2022. Available: https://www.frontiersin.org/journals/genetics/articles/10.3389/fgene.2022.1053559/full.

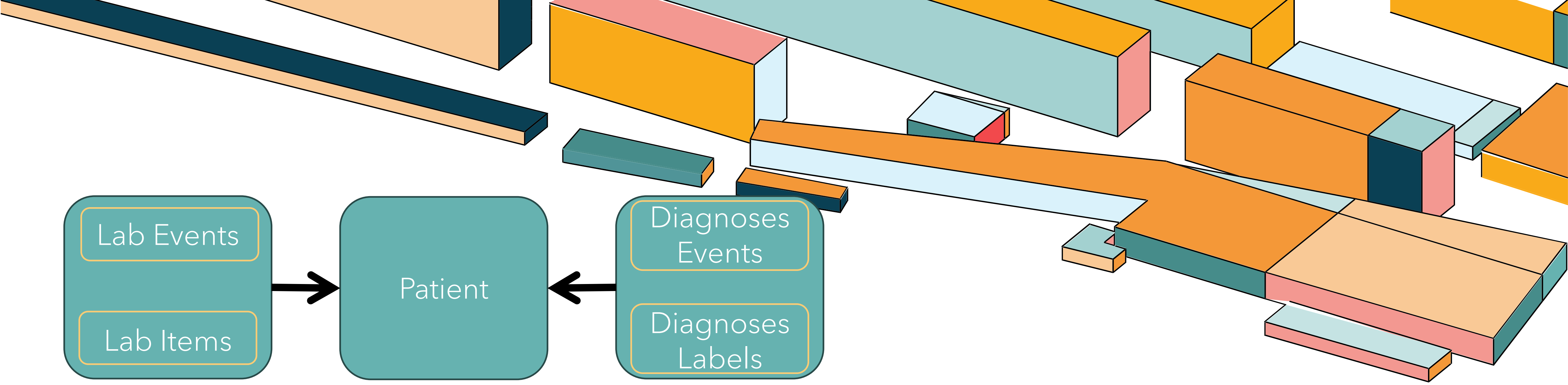
[5] P. Kováč, P. Jackuliak, A. Bražinová, I. Varga, M. Aláč, M. Smatana, D. Lovich, and A. Thurzo. "Artificial Intelligence-Driven Facial Image Analysis for the Early Detection of Rare Diseases: Legal, Ethical, Forensic, and Cybersecurity Considerations." AI, vol. 5, no. 3, pp. 990-1010, 2024. Available: https://www.mdpi.com/2673-2688/5/3/49.

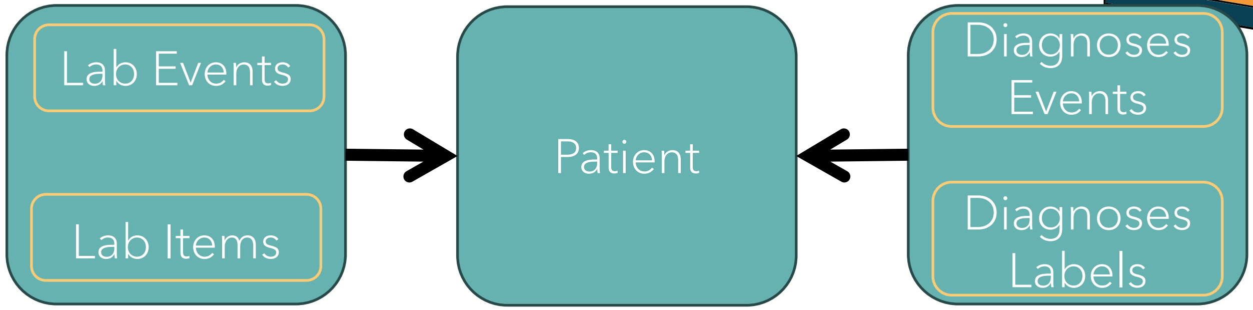
[6] M. Kaasgaard, K. Grebosz-Haring, C. Davies, G. Musgrave, J. Shriraam, J. M. McCrary, and S. Clift. "Is it premature to formulate recommendations for policy and practice, based on culture and health research? A robust critique of the CultureForHealth (2022) report." Front. Public Health, vol. 12, 2024. Available: https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2024.1373649/full.

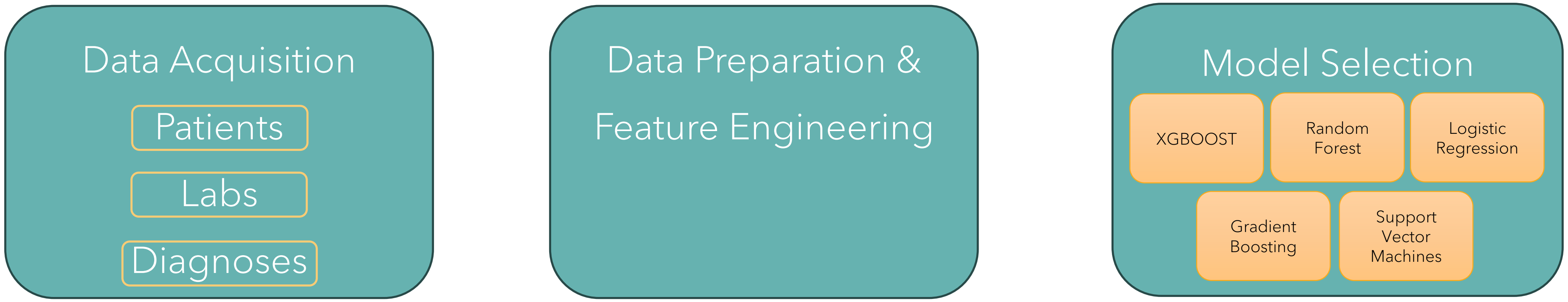
[7] FDA. "Rare Diseases at FDA." Available: https://www.fda.gov/patients/rare-diseases-fda.

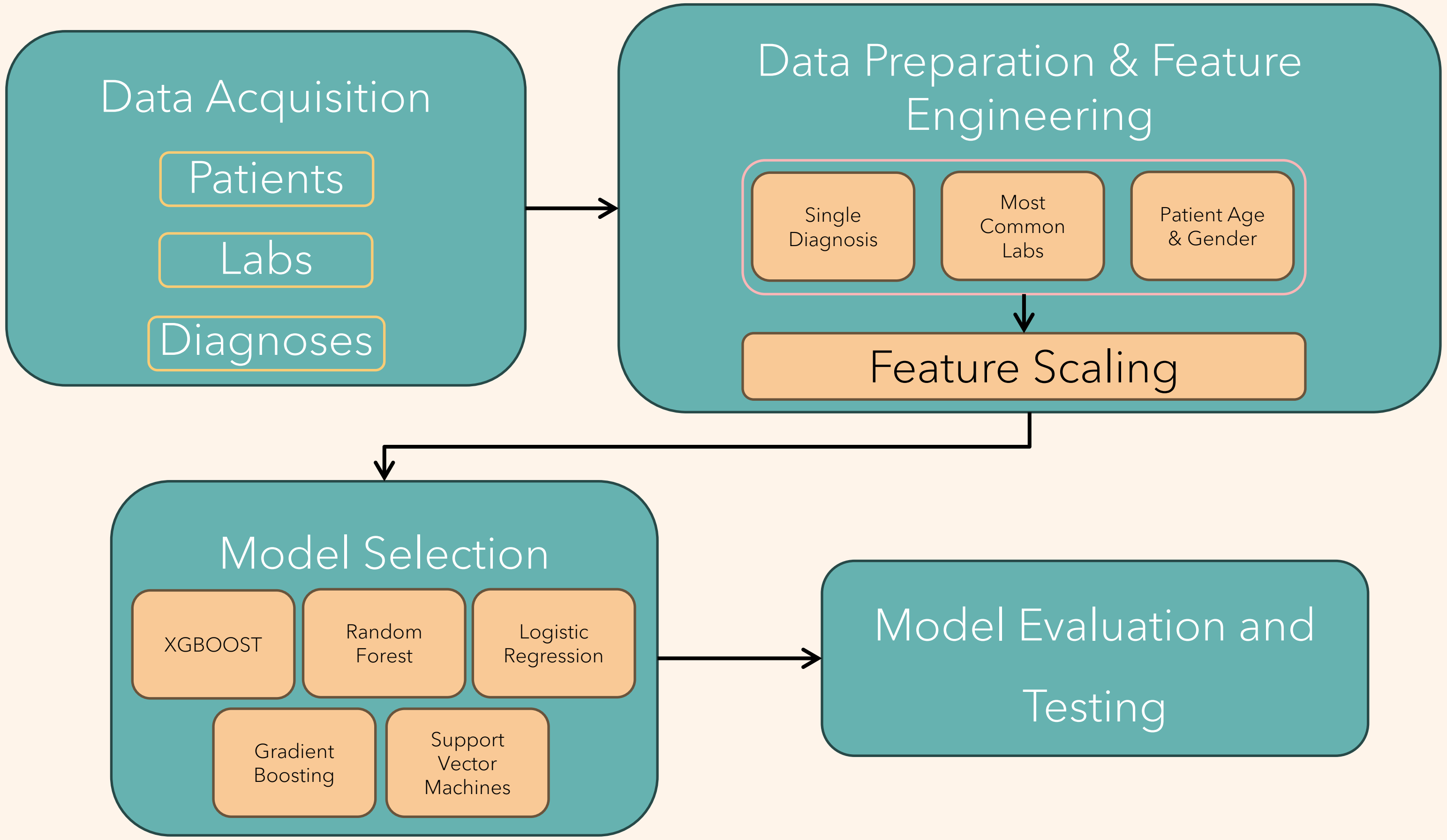
[8] Johnson, T. Pollard, and R. Mark. "MIMIC-III Clinical Database v1.4." PhysioNet, 2016. Available: https://physionet.org/content/mimiciii/1.4/.

[9] McMullen, Tara Hunt "Fighting for Those With Rare Diseases" Available: https://fightingfor.nd.edu/2024/fighting-for-those-with-rare-diseases/




Lab Events
Lab Items
Patient
Diagnoses Events
Diagnoses Labels


Data Acquisition
Patients
Labs
Diagnoses
Data Preparation &
Feature Engineering
Model Selection
XGBOOST
Random Forest
Logistic Regression
Gradient Boosting
Support Vector Machines



| Hyperparameter | Values |
|---|---|
| n_estimators | [200, 250] |
| max_depth | [10,12,14] |
| min_samples_split | [5, 8] |