USC Viterbi
School of Engineering

HOSPITAL READMISSION PIPELINE
FINAL REPORT
ISE-543 PROJECT

USC Viterbi School of Engineering

SAMPLE FINAL REPORT

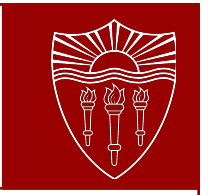
OUTLINE

- Business Objective
- Exploratory Data Analysis Report
- Data Preparation Plan
- Model Pipelines
- Summary Discussion

USC Viterbi School of Engineering

FINAL REPORT

BUSINESS OBJECTIVE

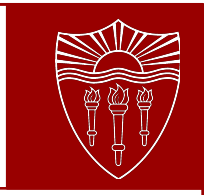


- Develop a model to predict whether patients would be readmitted to hospitals within 30 days after initial hospitalization using a healthcare readmission data set of anonymized patient information from three hospitals
- PatientID represent patients with different related information followed and the binary readmission results within 30 days
  - For binary readmission flag, value ‘1’ means this patient was readmitted, value ‘0’ means this patient was not readmitted
  - Patients information includes four categories: demographic information, lifestyle information, clinical information and utilization information

USC Viterbi
School of Engineering

EDA REPORT

OUTLINE

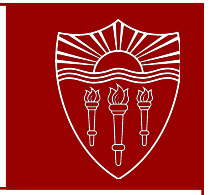


- Dataset overview
- Data quality summary
- Statistical summary of dataset
- Univariate analysis
- Bivariate analysis

USC Viterbi School of Engineering

EDA REPORT

DATASET OVERVIEW



- "healthcare_readmissions_train.csv" – dataset with 22 variables and 8038 observations
- Dataset contains one response variables:

Variable	Description
Readmission within 30 days	Binary variable (1 = patient was readmitted, 0 = patient was not readmitted)

USC Viterbi School of Engineering

DATASET OVERVIEW

DEMOGRAPHIC VARIABLES

Demographic variables describe the profile of each patient in terms of age, gender, and ethnicity

Variable	Description
Age	Patient age
Gender	String variable with values “Male” or “Female”
Ethnicity	String variable with values “Caucasian”, “Hispanic”, “African American” and “Other”

USC Viterbi School of Engineering

DATASET OVERVIEW

LIFESTYLE VARIABLES

Demographic variables describe the profile of each patient in terms of living situation, exercise frequency, and diet type

Variable	Description
Living Situation	Living arrangement: Lives Alone, Lives with Family, Assisted Living
Exercise Frequency	String variable with values “None”, “Occasional”, or “Regular”
Diet Type	String variable with values “Balanced”, “High-fat”, “Vegetarian”, “Other”

USC Viterbi School of Engineering

DATASET OVERVIEW

CLINICAL VARIABLES

Clinical variables reflects the patient's physical condition and disease condition, which is the medical diagnosis or health assessment variable.

Variable	Description
Height	Patient height in meters
Smoker	Boolean indicating if patient is a current smoker
BMI	Patient Body Mass Index
Weight	Patient weight in kg
Adjusted Weight	Health system-specific adjustments to patient weight (in kg)
Has Diabetes	Boolean indicating if patient has diabetes (0 or 1)
Has Hypertension	Boolean indicating if patient has hypertension (0 or 1)

USC Viterbi School of Engineering

DATASET OVERVIEW

UTILIZATION VARIABLES

Utilization Variables describes the frequency and manner in which patients use medical resources.

Variable	Description
Number of Prior Visits	Number of previous hospitalizations of the patient
Medications Prescribed	Number of different prescription medications patient is currently taking
Length of Stay	Length of the hospital stay in days
Type of Treatment	String variable with values “None”, “Minor Surgery”, “Major Surgery”, “Other Treatment”
Hospital ID	Unique identifier of hospitals

USC Viterbi School of Engineering

EDA REPORT

DATA QUALITY SUMMARY

Missing values:

	Missing Values	Percentage
Exercise Frequency	2350	29.236128
Number of Prior Visits	314	3.906444
Medications Prescribed	657	8.173675
Type of Treatment	2486	30.928092

<u>Four variables have missing values:</u>

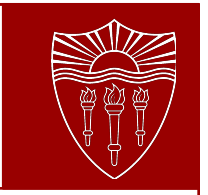
- Number of Prior Visits/ Medications Prescribed
  - Variables will be imputed by mean because the missing percentage is less than 30% and this variable is numerical

- Type of Treatment/ Exercise Frequency
  - Variable will be imputed by new value ‘missing value’ because the percentage is a little high(>30%) while being imputed by mode may cause information bias

USC Viterbi School of Engineering

EDA REPORT

DATA QUALITY SUMMARY



Data types:

```
Columns and Data Types:
Age                          int64
Gender                      object
Ethnicity                   object
Hospital ID                 object
Height (m)                 float64
Smoker                        bool
BMI                        float64
Weight (kg)                float64
Adjusted Weight (kg)       float64
Has Diabetes                 int64
Has Hypertension             int64
Exercise Frequency          object
Diet Type                   object
Number of Prior Visits     float64
Medications Prescribed     float64
Length of Stay               int64
Type of Treatment           object
Readmission within 30 Days   int64
dtype: object
```

Several string variables:

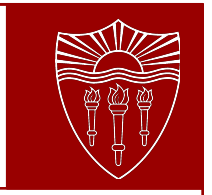
Categories that must be encoded before modeling:

- Gender
- Ethnicity
- Hospital ID
- Smoker
- Exercise Frequency
- Diet type
- Type of Treatment

USC Viterbi School of Engineering

DATASET STATISTICAL SUMMARY

CLINICAL VARIABLES



Descriptive statistics of clinical variables:

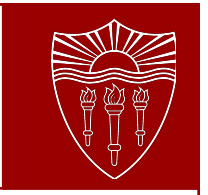
	Height (m)	BMI	Weight (kg)	Adjusted Weight (kg)	Has Diabetes	Has Hypertension
count	8038.000000	8038.000000	8038.000000	8038.000000	8038.000000	8038.000000
mean	1.700983	26.258335	77.145366	76.269064	0.132620	0.172431
std	0.104152	4.753106	18.961059	16.701363	0.339185	0.377778
min	1.300000	8.300000	23.300000	23.126324	0.000000	0.000000
25%	1.600000	23.100000	64.800000	64.720543	0.000000	0.000000
50%	1.700000	26.200000	75.400000	75.127095	0.000000	0.000000
75%	1.800000	29.300000	87.475000	86.904890	0.000000	0.000000
max	2.000000	44.000000	236.300000	159.051116	1.000000	1.000000

```
Smoker:

Smoker
False    6067
True     1971
Name: count, dtype: int64
```

USC Viterbi School of Engineering

DATASET STATISTICAL SUMMARY

UTILIZATION VARIABLES



Descriptive statistics of utilization variables:

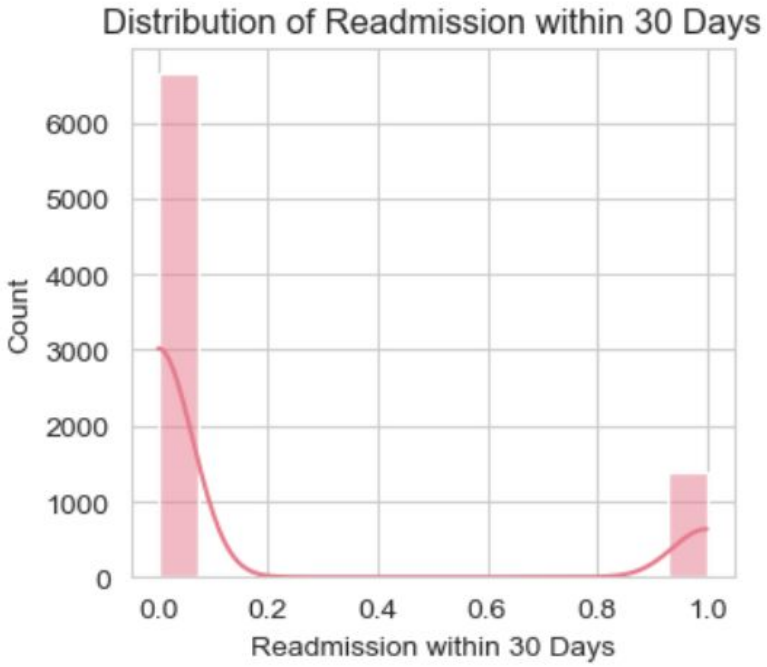
	Number of Prior Visits	Medications Prescribed	Length of Stay
count	7724.000000	7381.000000	8038.000000
mean	3.044795	3.509010	2.544041
std	1.740236	1.955142	2.970845
min	0.000000	0.000000	0.000000
25%	2.000000	2.000000	0.000000
50%	3.000000	3.000000	2.000000
75%	4.000000	5.000000	4.000000
max	11.000000	12.000000	23.000000

```
Type of Treatment:

Type of Treatment
Major Surgery      2415
Minor Surgery      1578
Other Treatment    1559
Name: count, dtype: int64
```

```
Hospital ID:

Hospital ID
Hosp1    2709
Hosp3    2673
Hosp2    2656
Name: count, dtype: int64
```

EDA REPORT – UNIVARIATE ANALYSIS

RESPONSE VARIABLES

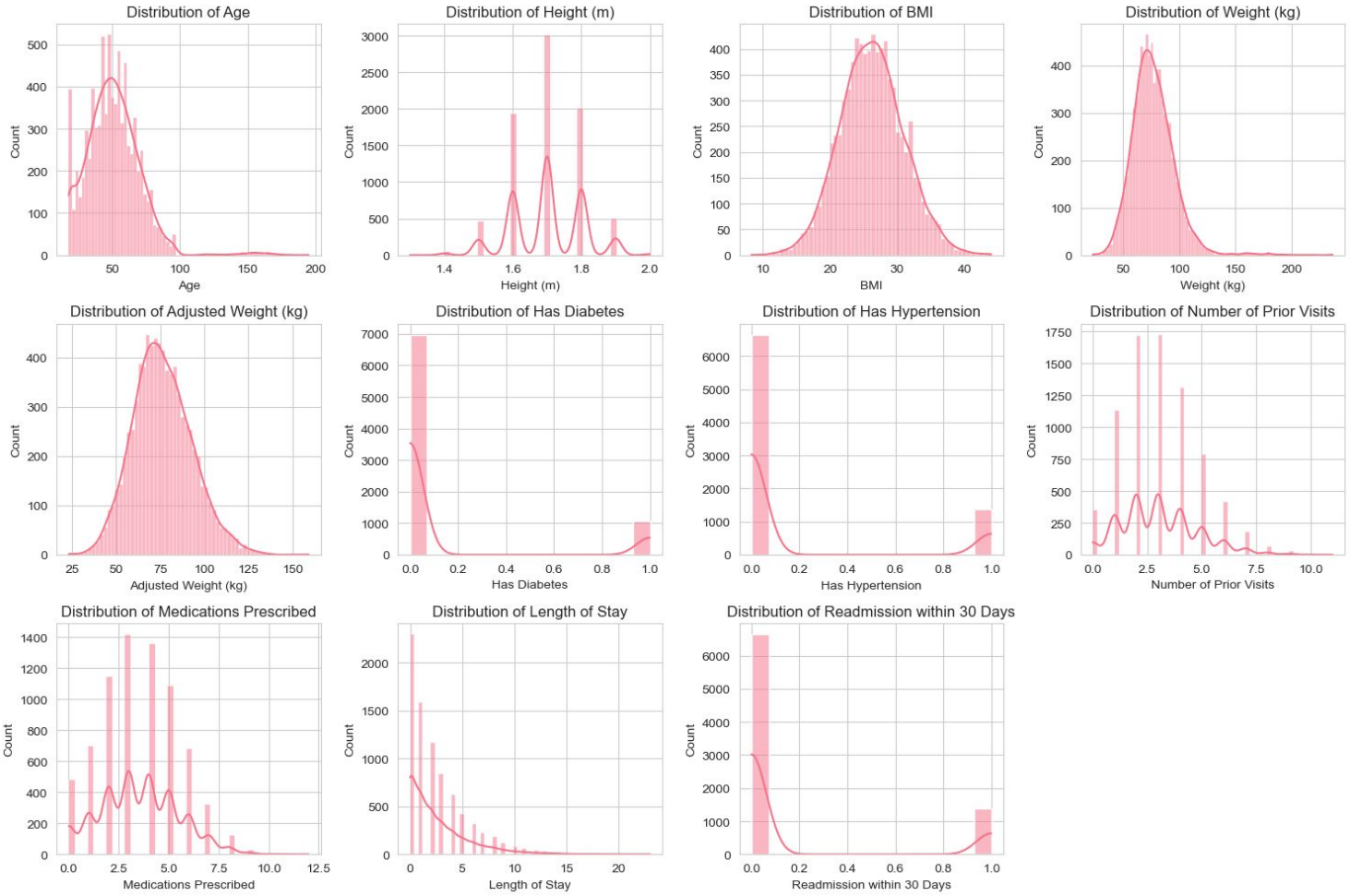


Response variable is moderately unbalanced
- Use SMOTE

USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

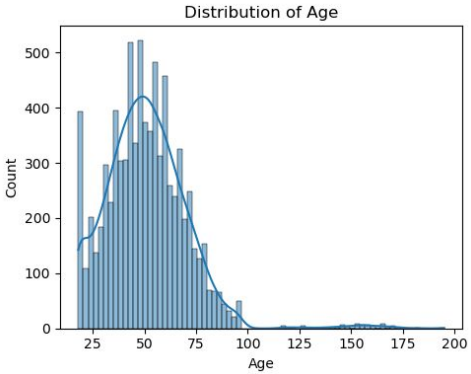
NUMERICAL OVERVIEW

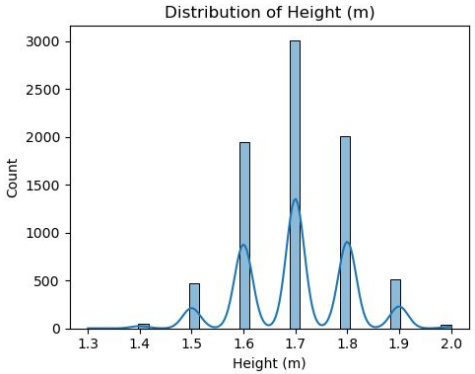


USC Viterbi School of Engineering

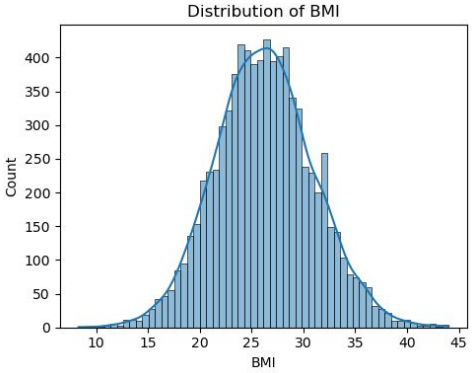
EDA REPORT – UNIVARIATE ANALYSIS

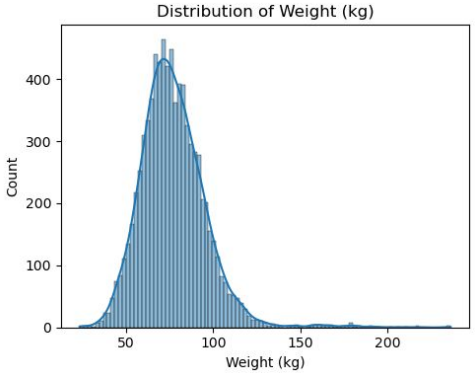
CLINICAL VARIABLES



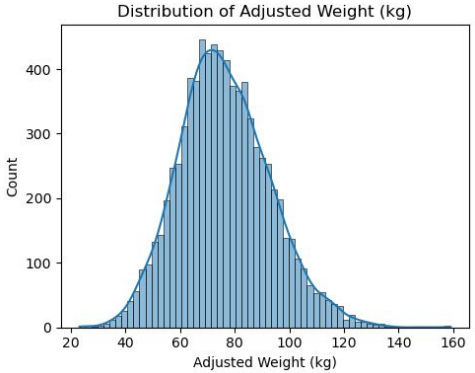


lightly right-skewed





slightly right-skewed



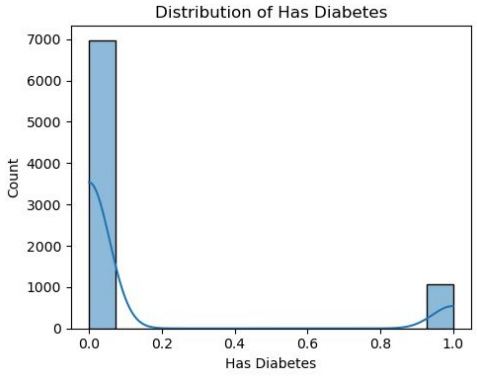
slightly right-skewed

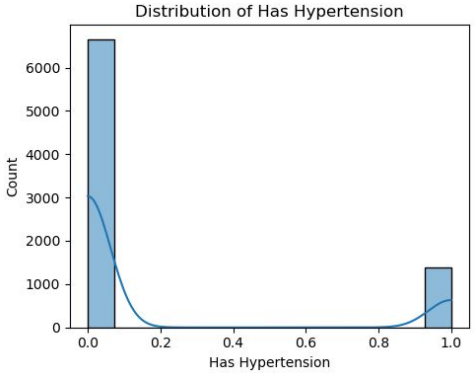
USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

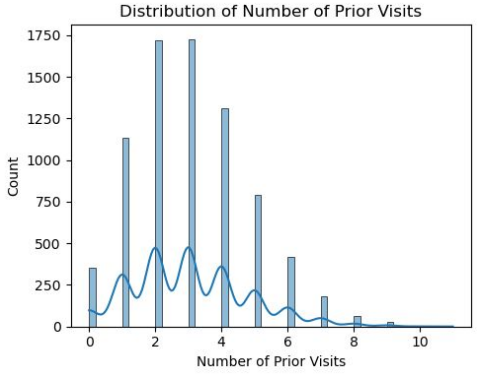
CLINICAL AND UTILIZATION VARIABLES

Clinical variables:

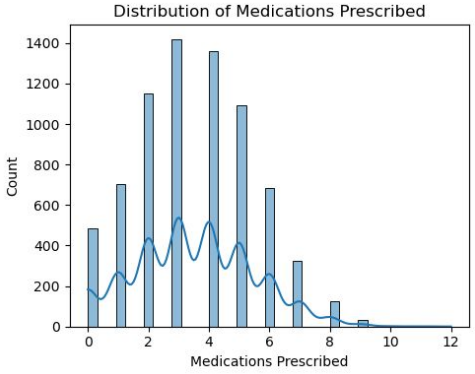




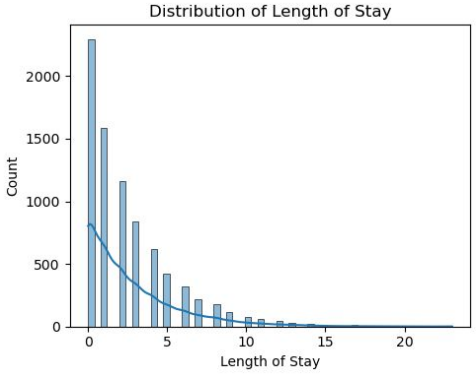
Utilization variables:



slightly right-skewed, use log transform



slightly right-skewed, use log transform

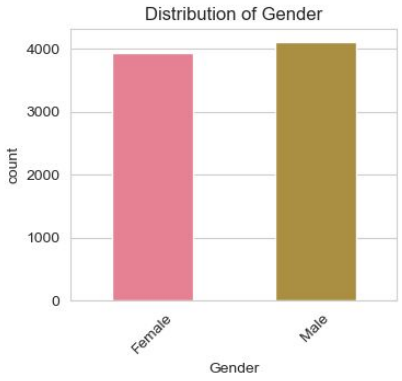


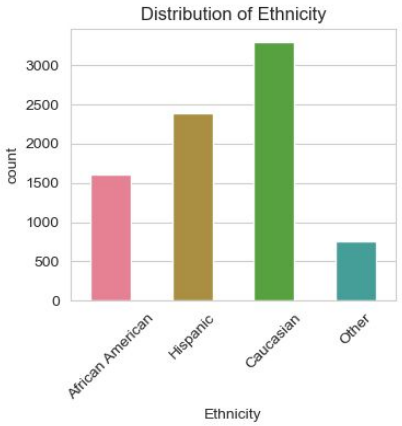
right-skewed, use log transform

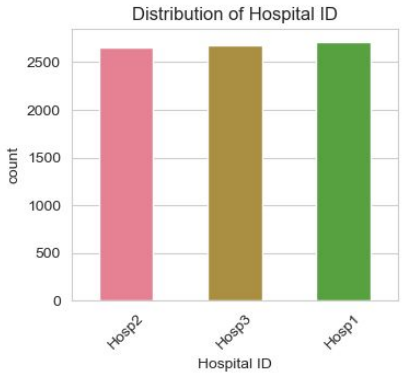
USC Viterbi School of Engineering

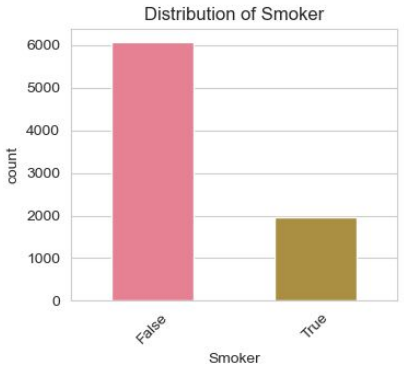
UNIVARIATE ANALYSIS

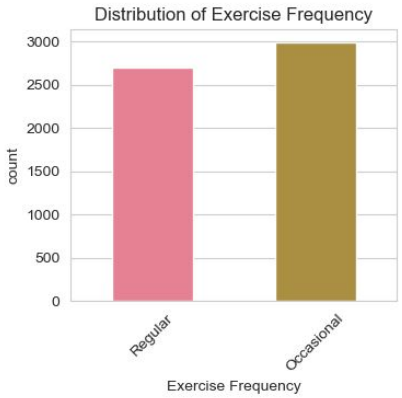
CATEGORICAL OVERVIEW

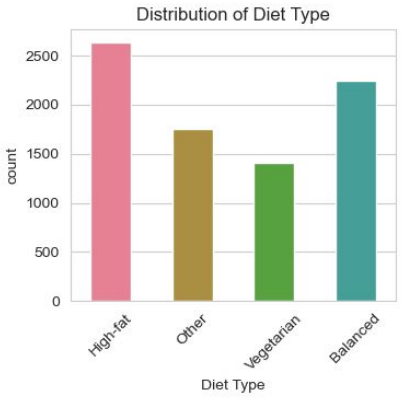


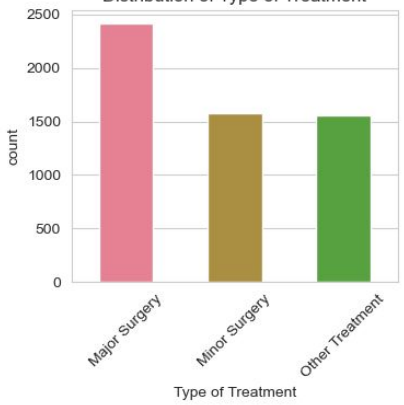










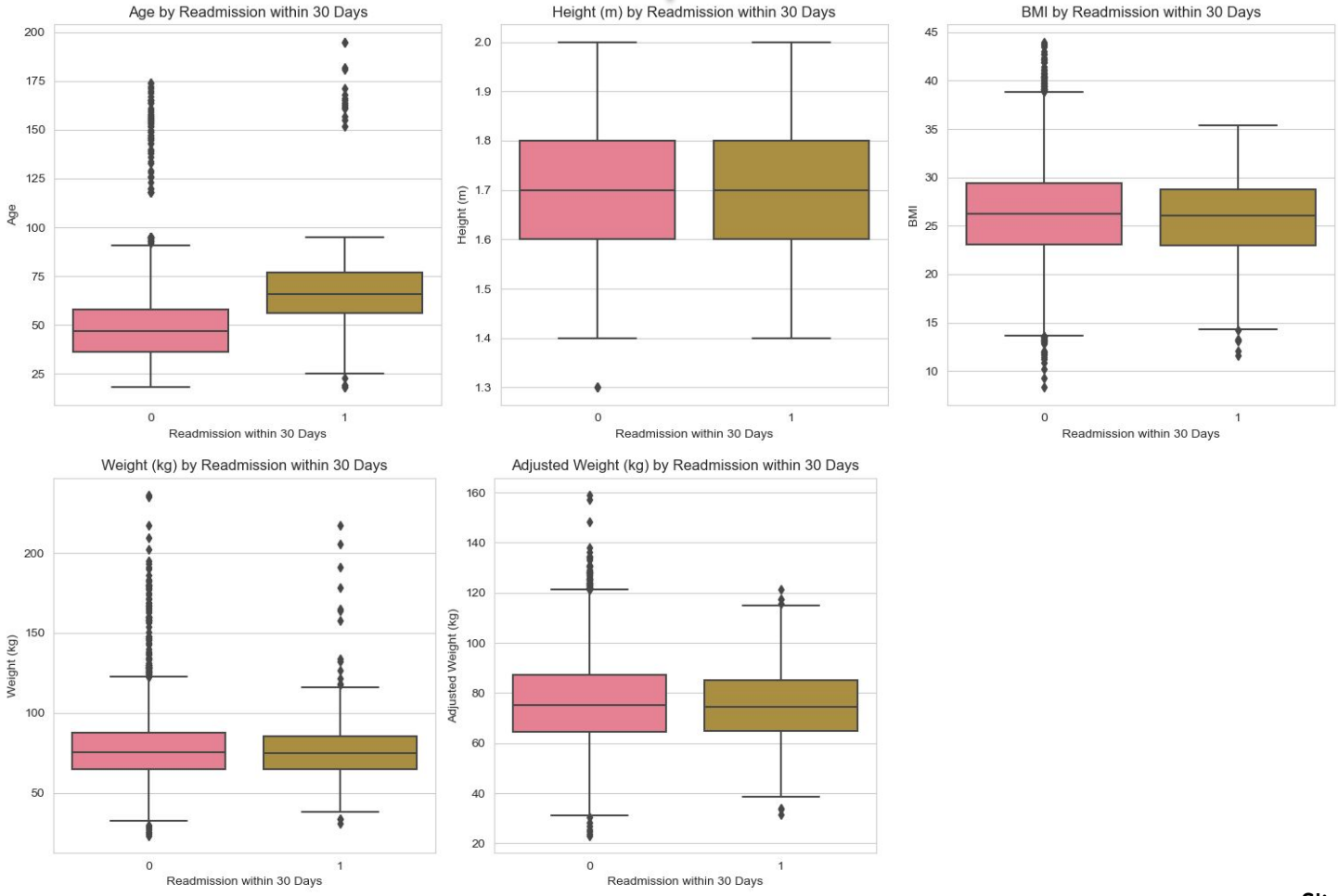


USC Viterbi School of Engineering

BIVARIATE ANALYSIS

CLINICAL VARIABLES VS RESPONSE

No significant correlation with the target variable

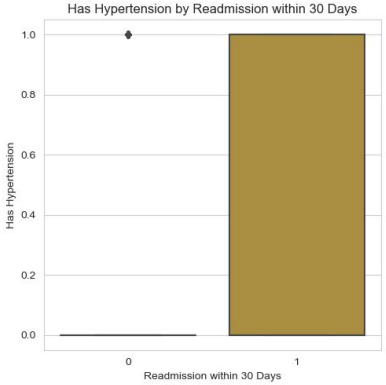


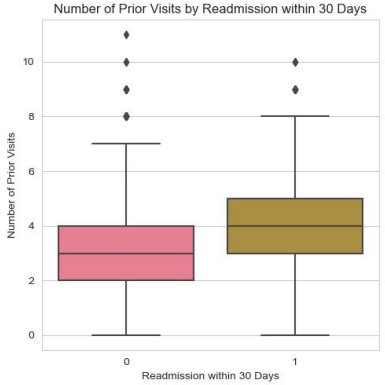
USC Viterbi School of Engineering

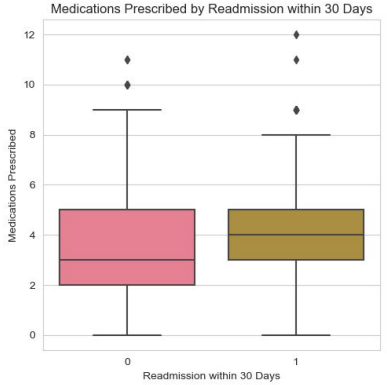
BIVARIATE ANALYSIS

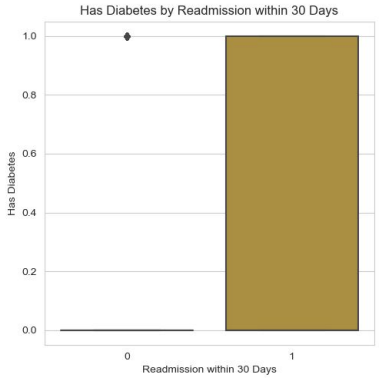
CLINICAL VARIABLES VS RESPONSE

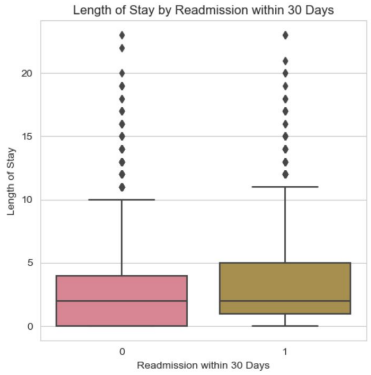
very correlated with the target variable











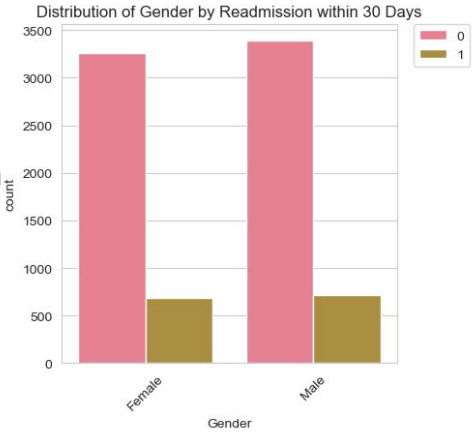
USC Viterbi School of Engineering

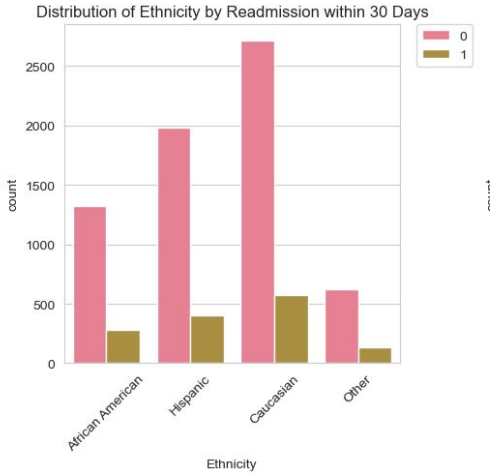
BIVARIATE ANALYSIS

DEMOGRAPHIC & CLINICAL VS RESPONSE

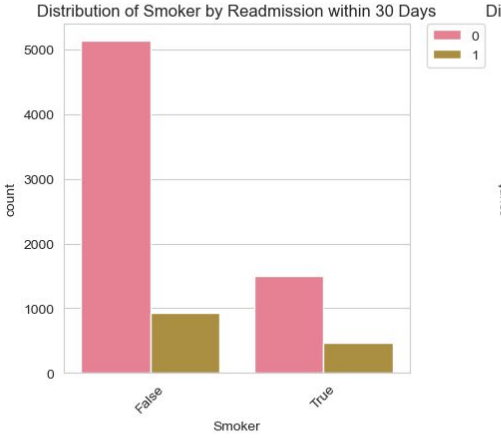
Demographic Variables

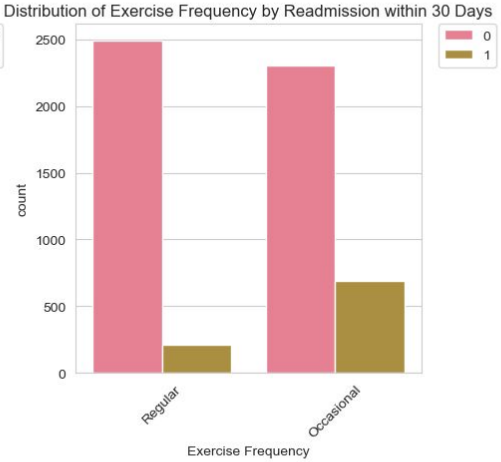
No significant correlation with the target variable

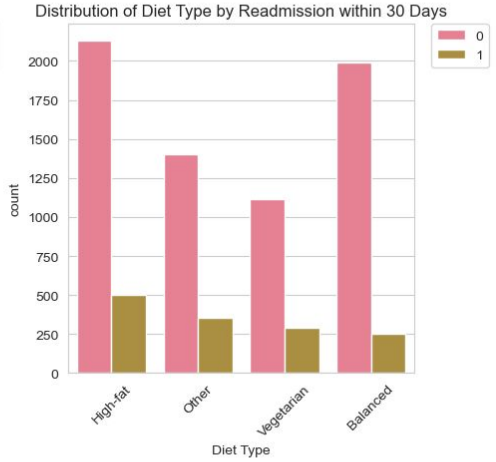




Clinical variables



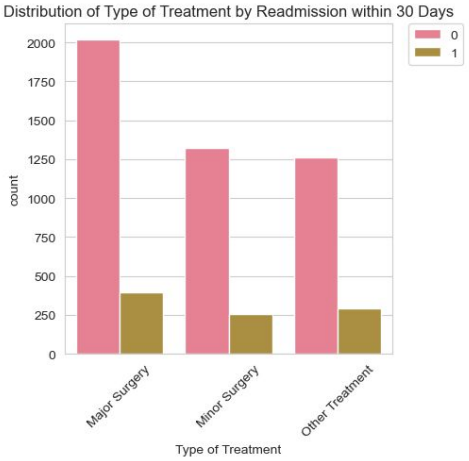


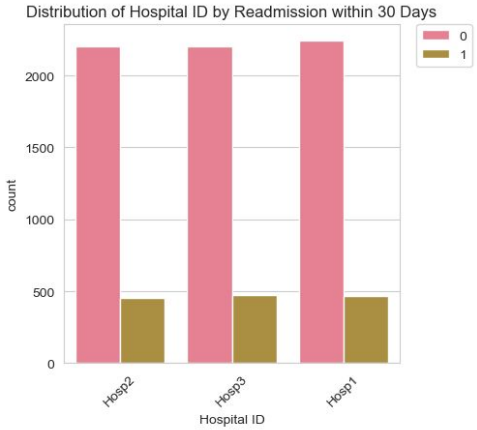


USC Viterbi School of Engineering

BIVARIATE ANALYSIS

UTILIZATION VS RESPONSE



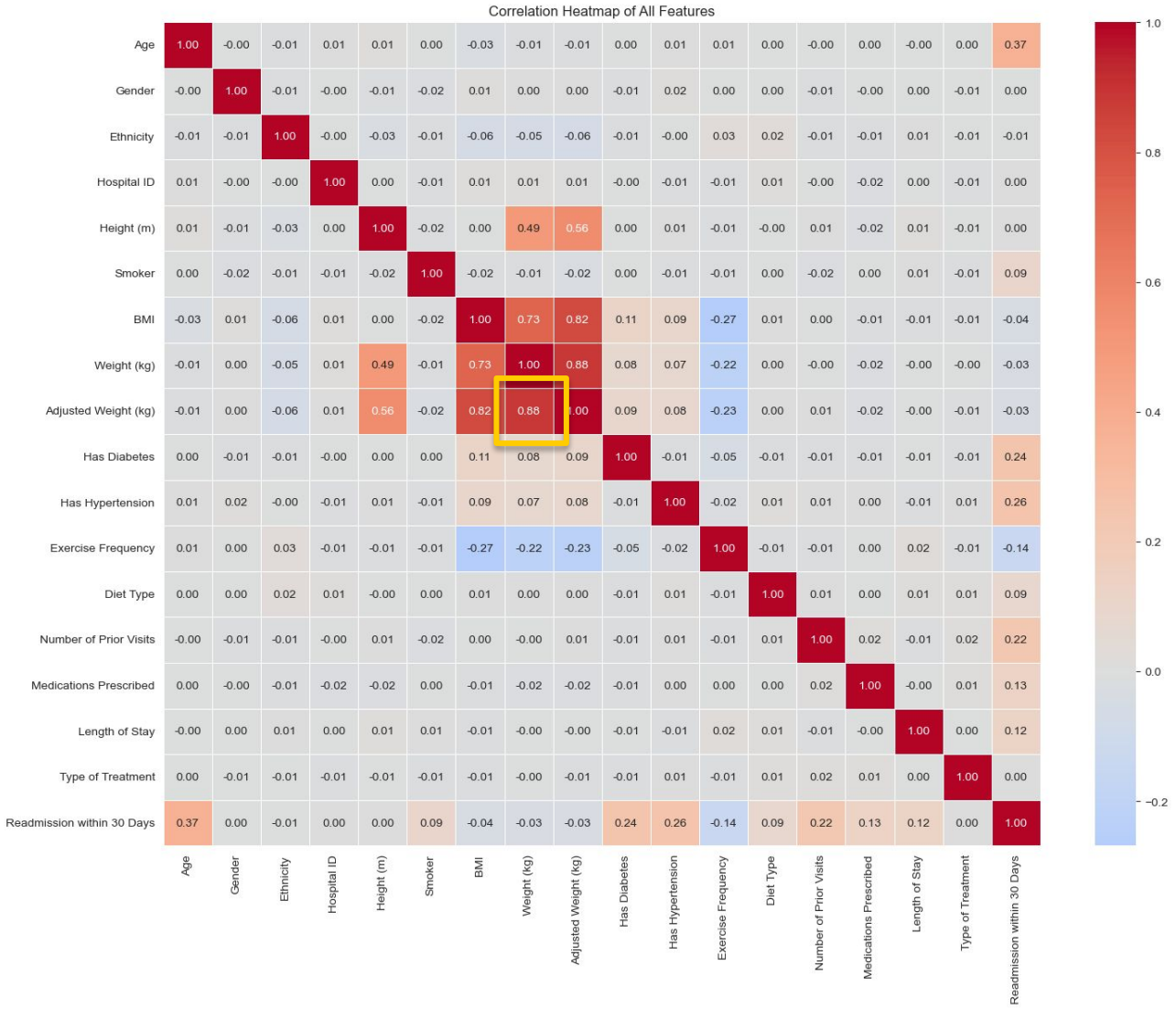


No significant correlation with the target variable

USC Viterbi School of Engineering

BIVARIATE ANALYSIS

OTHER VARIABLES VS RESPONSE



Weight (kg) and Adjusted Weight (kg) are highly correlated, maybe redundant variables

USC Viterbi School of Engineering

DATA PREPARATION PLAN

OVERVIEW

- Data quality issues and actions
- Feature engineering decisions
- Feature selection decisions
- Dataset partitioning decisions
- Final dataset summary

DATA PREPARATION PLAN

DATA QUALITY ISSUES AND ACTIONS

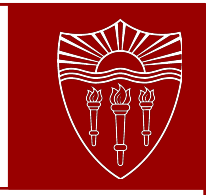
Missing variables

- Exercise Frequency/ Number of Prior Visits/ Medications Prescribed
  - Variables will be imputed by mean because the missing percentage is less than 30% and this variable is numerical

- Type of Treatment
  - Variable will be imputed by new value ‘missing value’ because the percentage is a little high(>30%) while being imputed by mode may cause information bias

DATA PREPARATION PLAN

FEATURE ENGINEERING DECISIONS

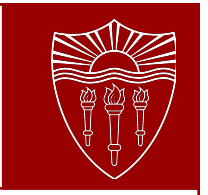


- Log-transform Length of Stay, Medications Prescribed, and Number of Prior Visits due to strong positive skew
- Encode Gender Ethnicity, Hospital ID, Smoker, Exercise Frequency, Diet type, and Type of Treatment by label encoding
- Scale all variables after encoding by StandardScaler

USC Viterbi School of Engineering

DATA PREPARATION PLAN

FEATURE SELECTION DECISIONS



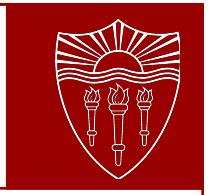
- Originally drop Weight (kg) because it highly correlated with Adjusted Weight (kg)
  - correlation coefficient = 0.88
- but experiment results turns out that model performs better when keeping Weight (kg) variable: higher f1 score and higher accuracy

Experiment Model	F1 Score	ROC AUC
XGBoost with Weight (kg)	0.765	0.72
XGBoost without Weight (kg)	0.750	0.71

- Therefore, no variables are decided to be dropped

DATA PREPARATION PLAN

DATASET PARTITIONING DECISIONS



- Perform SMOTE oversampling due to unbalanced dataset

```
Class distribution after SMOTE:
Readmission within 30 Days
0    5287
1    5287
Name: count, dtype: int64
```

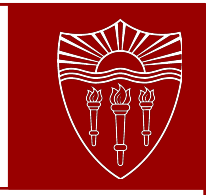
- After partitioning and SMOTE: 12181 rows x 18 columns in total

	X_train	y_train	X_val	y_val
rows	10574	10574	1608	1608
columns	17	1	17	1

USC Viterbi School of Engineering

DATA PREPARATION PLAN

FINAL DATASET SUMMARY



Variable	Description	Preprocess and feature engineering
PatientID	Unique patient identifier	Unique identifier
Age	Patient age	Numerical, imputed by mean
Gender	String variable with values “Male” or “Female”	Categorical
Ethnicity	String variable with values “Caucasian”, “Hispanic”, “African American” and “Other”	Categorical
Hospital ID	Hospital identifier with values “Hosp1”, “Hosp2”, and “Hosp3”	Categorical, imputed by new category ‘missing value’
Height	Patient height in meters	Numerical, imputed by mean
Smoker	Boolean indicating if patient is a current smoker	Categorical, imputed by new category ‘missing value’
BMI	Patient Body Mass Index	Numerical
Weight	Patient weight in kg	Numerical
Adjusted Weight	Health system-specific adjustments to patient weight (in kg)	Numerical

USC Viterbi School of Engineering

DATA PREPARATION PLAN

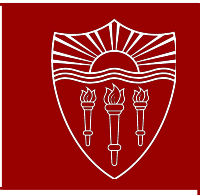
FINAL DATASET SUMMARY

Variable	Description	Preprocess and feature engineering
Has Diabetes	Boolean indicating if patient has diabetes (0 or 1)	Numerical
Has Hypertension	Boolean indicating if patient has hypertension (0 or 1)	Numerical
Has Chronic Kidney Disease	Boolean indicating if patient has chronic kidney disease (0 or 1)	Numerical
Living Situation	Living arrangement: Lives Alone, Lives with Family, Assisted Living	Categorical
Exercise Frequency	String variable with values “None”, “Occasional”, or “Regular”	Categorical, imputed by new category ‘missing value’
Diet Type	String variable with values “Balanced”, “High-fat”, “Vegetarian”, “Other”	Numerical
Number of Prior Visits	Number of previous hospitalizations of the patient	Numerical, imputed by mean, log transform
Medications Prescribed	Number of different prescription medications patient is currently taking	Numerical, imputed by mean, log transform
Length of Stay	Length of the hospital stay in days	Numerical, log transform
Type of Treatment	String variable with values “None”, “Minor Surgery”, “Major Surgery”, “Other Treatment”	Categorical, imputed by new category ‘missing value’
Readmission within 30 days	Binary variable (1 = patient was readmitted, 0 = patient was not readmitted)	Response variable

USC Viterbi School of Engineering

MODEL DEVELOPING PLAN

MODEL SELECTION



- Different models and hyperparameter set were tried for model selection
- Model performance

Model	F1 Score
Logistic Model	0.765
Random Forest	0.750
XGBoost with hyperparameter group 1	0.800
XGBoost with hyperparameter group 2	0.790
Neural Network	0.755

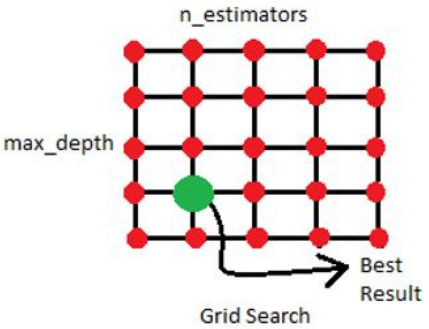
USC Viterbi School of Engineering

MODEL DEVELOPING PLAN

XGBOOST HYPERPARAMETER TUNING

- Conduct hyperparameter tuning by gridsearch, and find out best hyperparameters for XGBoost model
- Hyperparameters:
  - 'colsample_bytree': 1.0,
  - 'learning_rate': 0.1,
  - 'max_depth': 9,
  - 'n_estimators': 200,
  - 'subsample': 0.8

```
xgb_param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [9, 7],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
```



XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, feature_weights=None, gamma=0,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=6, max_leaves=None,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=200, n_jobs=-1,
              num_parallel_tree=None, ...)
```

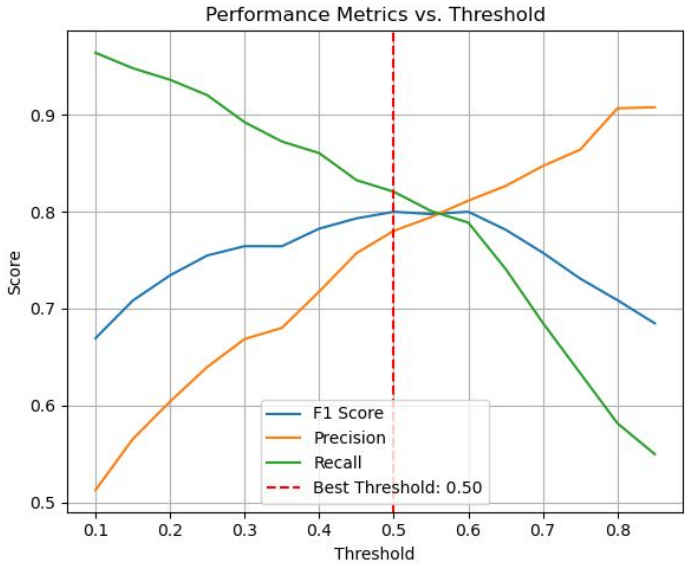
USC Viterbi School of Engineering

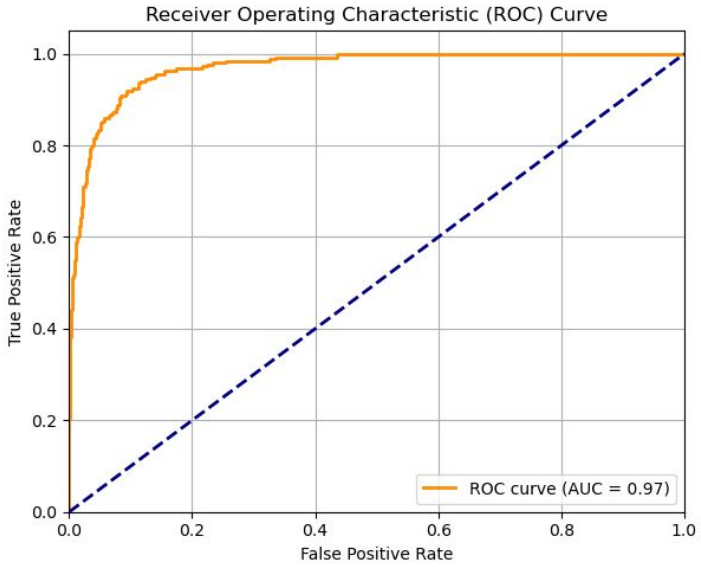
MODEL DEVELOPING PLAN

XGBOOST THRESHOLD SELECTION

- By experiment, the default threshold 0.5 is the best threshold. Therefore, we use this threshold for the following model and pipeline

```
Best threshold: 0.50
Best F1 score: 0.8000
Corresponding precision: 0.7803
Corresponding recall: 0.8207
```





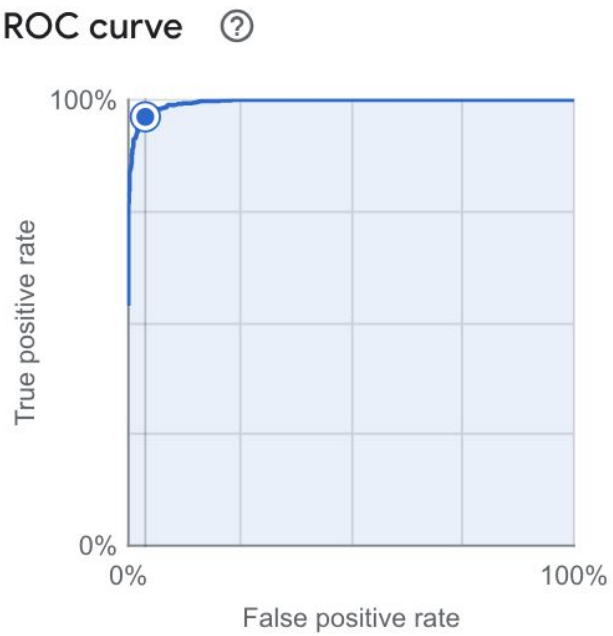
USC Viterbi School of Engineering

MODEL DEVELOPING PLAN

AUTOML MODEL STRUCTURE

- AutoML model is also tried for this prediction, without any data preprocessing and feature engineering, AutoML created an ensemble model with 25 components:
  - 14 Neural Network models
  - 11 boosted tree models
  - Achieve 0.96 micro-average F1 score

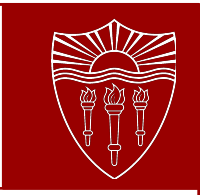
PR AUC	0.995
ROC AUC	0.995
Log loss	0.101
Micro-average F1	0.9637174
Macro-average F1	0.94080365
Micro-average precision	96.7%
Micro-average recall	96.1%
Total items	0
Training items	0
Validation items	0
Test items	0



USC Viterbi School of Engineering

MODEL DEVELOPING PLAN

AUTOML MODEL PERFORMANCE



- AutoML model is also tried for this prediction, achieving better F1_Score than all other manually tuned models
- AutoML model achieves 0.8814 F1 Score on the test dataset

1) F1-Score (88.15/100)

```
Test Failed: 88.15 != 100 : The F1-Score is 0.8814589665653495.
```

- Manually tuned model achieves 0.7658 F1 Score on the test dataset

1) F1-Score (76.58/100)

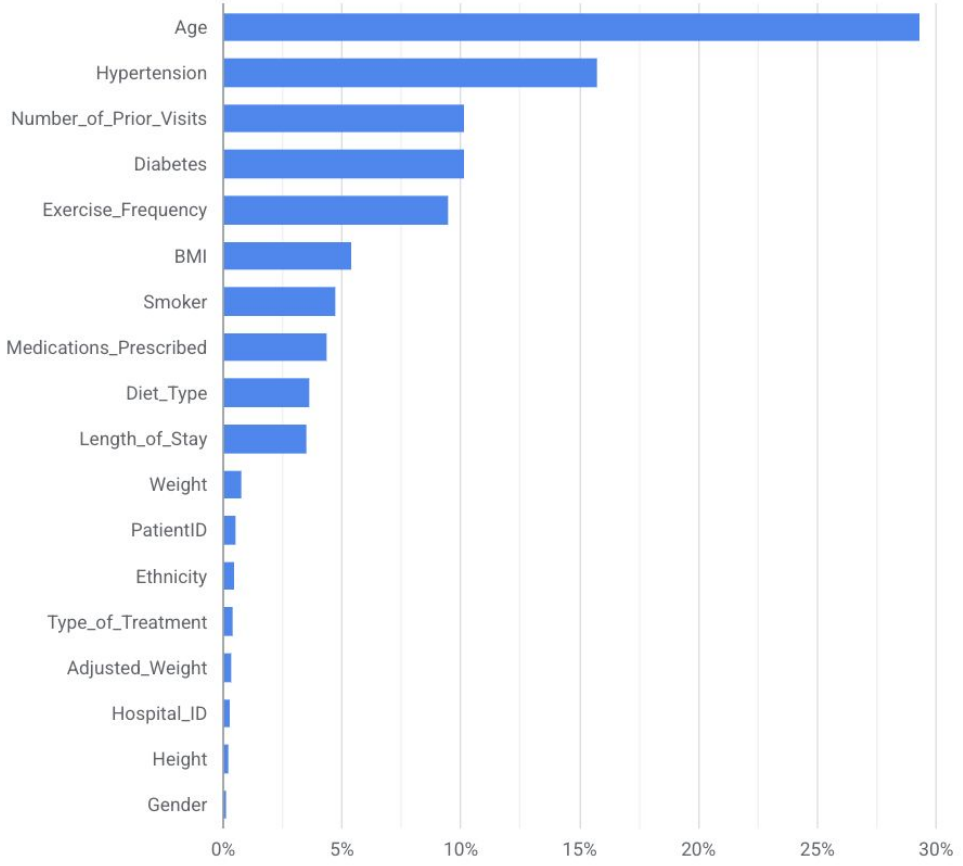
```
Test Failed: 76.58 != 100 : The F1-Score is 0.7658321060382917.
```

- For the next pipeline part, we still use the manually tuned model to display a complete pipeline building process

USC Viterbi School of Engineering

MODEL DEVELOPMENT

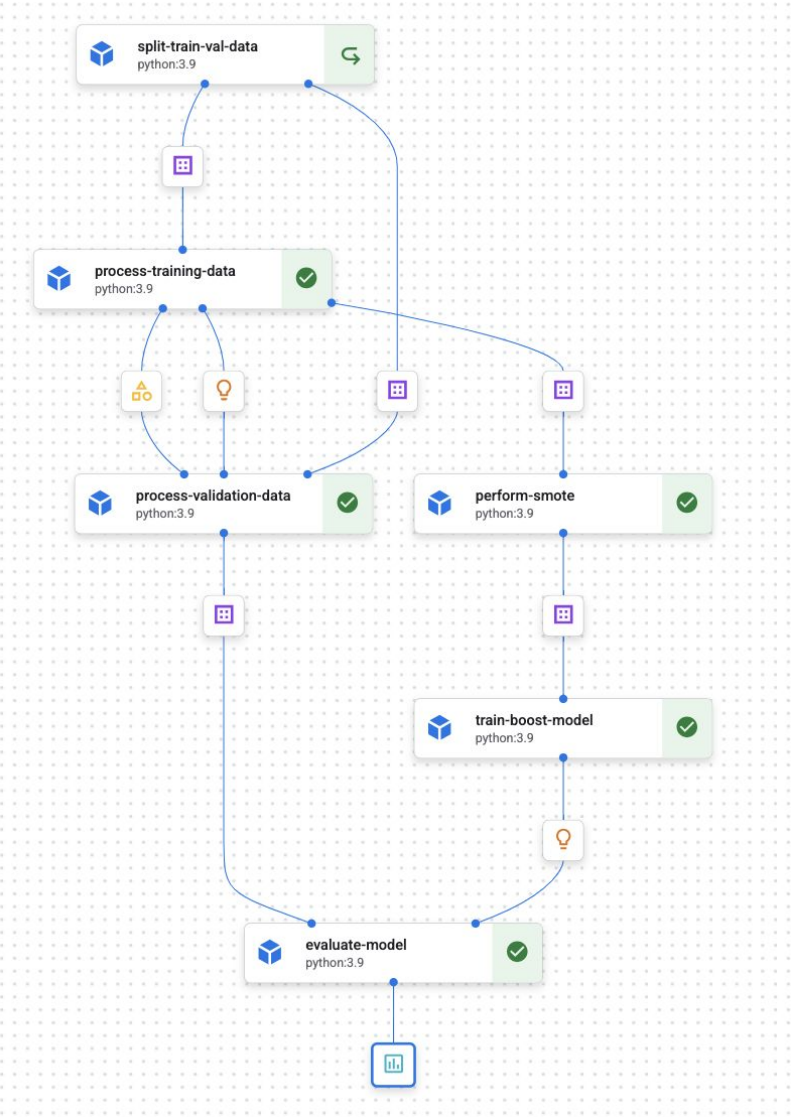
AUTOML MODEL FEATURE IMPORTANCE



USC Viterbi School of Engineering

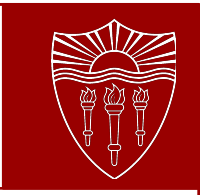
PIPELINE DESIGN

TRAINING PIPELINE



TRAINING PIPELINE

PIPELINE DEFINITION CODE

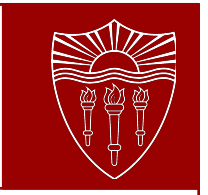


```python
@component(packages_to_install=['pandas', 'scikit-learn', 'fsspec', 'gcsfs'])
def split_train_val_data(
    input_data_path: str,
    train_dataset: Output[Dataset],
    val_dataset: Output[Dataset]
):
    import pandas as pd
    from sklearn.model_selection import train_test_split

    df = pd.read_csv(input_data_path)
    train_df, val_df = train_test_split(df, test_size=0.2, random_state=42)

    train_df.to_csv(train_dataset.path, index=False)
    val_df.to_csv(val_dataset.path, index=False)
```

USC Viterbi School of Engineering

TRAINING PIPELINE

PERFORM_INITIAL_DATA_PREPARATION



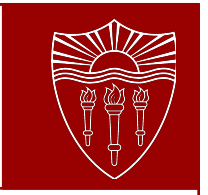
```python
from kfp.v2.dsl import component, Input, Output, Dataset, Model

@component(
    packages_to_install=["pandas", "scikit-learn", "joblib"],
)
def process_training_data(
    train_dataset: Input[Dataset],
    processed_train_dataset: Output[Dataset],
    scaler_model: Output[Model],
    all_mean: Output[Artifact],
):
    import pandas as pd
    import numpy as np
    import joblib
    from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder

    # Load training dataset
    df = pd.read_csv(train_dataset.path)

    # Separate features and target column
    ids = df['PatientID']
    X_train = df.drop(columns=['PatientID', 'Readmission within 30 Days'])
    y_train = df["Readmission within 30 Days"]

    # Missing values
    numeric_cols = X_train.select_dtypes(include=['int64', 'float64']).columns
    categorical_cols = X_train.select_dtypes(include=['object', 'bool']).columns
    for col in numeric_cols:
      mean_value = X_train[col].mean()
      X_train[col] = X_train[col].fillna(mean_value)
    for col in categorical_cols:
      X_train[col] = X_train[col].fillna('missing')

    # Encode categorical variables
    categorical_cols = X_train.select_dtypes(include=['object', 'bool']).columns
    label_encoders = {}
    for col in categorical_cols:
        label_encoders[col] = LabelEncoder()
        X_train[col] = label_encoders[col].fit_transform(X_train[col])
```

```python
# Log-transform skewed features
X_train['Length of Stay_log'] = np.log1p(X_train['Length of Stay'])
X_train['Medications Prescribed_log'] = np.log1p(X_train['Medications Prescribed'])
X_train['Number of Prior Visits_log'] = np.log1p(X_train['Number of Prior Visits'])

# Fit StandardScaler on training features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

# Reconstruct normalized dataframe
X_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns)

# Concat
scaled_df = pd.concat([X_scaled_df, y_train], axis=1)

# Save the scaled training dataset
scaled_df.to_csv(processed_train_dataset.path, index=False)

# Save the fitted scaler for future use
joblib.dump(scaler, scaler_model.path)

# Log the median value in artifact metadata
all_mean.metadata['value'] = mean_value

# calculate mean of every column and save to metadata
for col in numeric_cols:
    mean_values = X_train[numeric_cols].mean().to_dict()
for key, value in mean_values.items():
    all_mean.metadata[key] = float(value)
```

USC Viterbi School of Engineering

TRAINING PIPELINE

PERFORM_INITIAL_DATA_PREPARATION



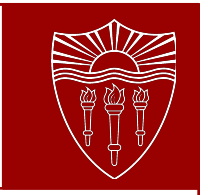
```python
from kfp.v2.dsl import component, Input, Output, Dataset, Model

@component(
    packages_to_install=["pandas", "scikit-learn", "joblib"],
)
def process_validation_data(
    val_dataset: Input[Dataset],
    scaler_model: Input[Model],
    all_mean: Input[Artifact],
    processed_val_dataset: Output[Dataset],
):
    import pandas as pd
    import joblib
    import numpy as np
    from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder

    # Load validation dataset
    df = pd.read_csv(val_dataset.path)

    # Separate features and target column
    ids = df['PatientID']
    X_val = df.drop(columns=['PatientID', 'Readmission within 30 Days'])
    y_val = df["Readmission within 30 Days"]

    # Missing values
    numeric_cols = X_val.select_dtypes(include=['int64', 'float64']).columns
    categorical_cols = X_val.select_dtypes(include=['object', 'bool']).columns
    for col in numeric_cols:
      X_val[col] = X_val[col].fillna(all_mean.metadata['value'], inplace=True)
    for col in categorical_cols:
      X_val[col] = X_val[col].fillna('missing')

    # Encode categorical variables
    categorical_cols = X_val.select_dtypes(include=['object', 'bool']).columns
    label_encoders = {}
    for col in categorical_cols:
        label_encoders[col] = LabelEncoder()
        X_val[col] = label_encoders[col].fit_transform(X_val[col])
```

```python
# Log-transform skewed features
X_val['Length of Stay_log'] = np.log1p(X_val['Length of Stay'])
X_val['Medications Prescribed_log'] = np.log1p(X_val['Medications Prescribed'])
X_val['Number of Prior Visits_log'] = np.log1p(X_val['Number of Prior Visits'])

# Load the scaler trained on the training set
scaler = joblib.load(scaler_model.path)

# Apply the scaler to validation features
X_scaled = scaler.transform(X_val)

# Reconstruct normalized dataframe
X_scaled_df = pd.DataFrame(X_scaled, columns=X_val.columns)

# Concat
scaled_df = pd.concat([X_scaled_df, y_val], axis=1)

# Save the normalized validation dataset
scaled_df.to_csv(processed_val_dataset.path, index=False)
```

USC Viterbi School of Engineering

TRAINING PIPELINE

IMPUTE_AGE_TRAINING

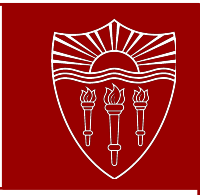


```python
@component(packages_to_install=[
    "pandas==1.4.4",
    "numpy==1.21.6",
    "scikit-learn==1.1.3",
    "imbalanced-learn==0.10.1"
])
def perform_SMOTE(
    input_df: Input[Dataset],
    output_df: Output[Dataset]
):
    import pandas as pd
    import numpy as np
    from imblearn.over_sampling import SMOTE

    # Load the input dataset
    df = pd.read_csv(input_df.path)

    # Separate features and target column
    X_train = df.drop(columns=['Readmission within 30 Days'])
    y_train = df["Readmission within 30 Days"]


    # Perform SMOTE oversampling
    smote = SMOTE()
    X_smote, y_smote = smote.fit_resample(X_train, y_train)

    # Recombine features and target
    X_smote_df = pd.DataFrame(X_smote, columns=X_train.columns)
    y_smote_df = pd.DataFrame(y_smote, columns=['Readmission within 30 Days'])
    oversampled_df = pd.concat([X_smote_df, y_smote_df], axis=1)

    # Save the oversampled dataset
    oversampled_df.to_csv(output_df.path, index=False)
```

USC Viterbi School of Engineering

TRAINING PIPELINE

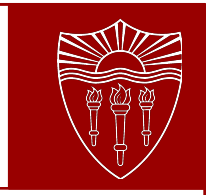
IMPUTE_AGE_VALIDATION



```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib", "xgboost", "scikit-learn"])
def train_boost_model(
    training_dataset: Input[Dataset],
    trained_model: Output[Model]
):
    import pandas as pd
    import joblib
    import xgboost as xgb
    from sklearn.model_selection import GridSearchCV

    # Load the training data
    train_df = pd.read_csv(training_dataset.path)

    # Split features and target
    X_train = train_df.drop('Readmission within 30 Days', axis=1)
    y_train = train_df['Readmission within 30 Days']

    # Define parameter grid for XGBoost
    xgb_param_grid = {
        'n_estimators': [100, 200],
        'max_depth': [9, 7],
        'learning_rate': [0.01, 0.1],
        'subsample': [0.8, 1.0],
        'colsample_bytree': [0.8, 1.0]
    }

    # Create XGBoost model
    xgb_model = xgb.XGBClassifier(
        random_state=42,
        eval_metric='logloss'
    )

    # Create GridSearchCV object
    xgb_grid_search = GridSearchCV(
        estimator=xgb_model,
        param_grid=xgb_param_grid,
        cv=5,
        scoring='f1',
        n_jobs=1,
        verbose=2
    )

    # Train the model
    model = xgb_grid_search.fit(X_train, y_train)

    # Save trained model to output path
    joblib.dump(model.best_estimator_, trained_model.path)
```

TRAINING PIPELINE

CLUSTERING - TRAINING

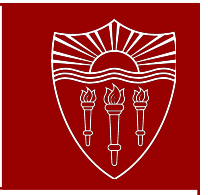


```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib", "xgboost"])
def evaluate_model(
    test_dataset: Input[Dataset],
    trained_model: Input[Model],
    metrics: Output[Metrics]
):
    import pandas as pd
    import joblib
    from sklearn.metrics import confusion_matrix, accuracy_score, f1_score

    # Load test dataset
    test_df = pd.read_csv(test_dataset.path)
    X_test = test_df.drop(columns=['Readmission within 30 Days'])
    y_test = test_df['Readmission within 30 Days']

    # Load the trained model
    model = joblib.load(trained_model.path)

    # Make predictions
    y_pred = model.predict(X_test)

    # Confusion matrix and metrics
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')

    # Log metrics for UI visibility
    metrics.log_metric("accuracy", accuracy)
    metrics.log_metric("f1_score", f1)
    metrics.log_metric("true_negatives", int(tn))
    metrics.log_metric("false_positives", int(fp))
    metrics.log_metric("false_negatives", int(fn))
    metrics.log_metric("true_positives", int(tp))
```

USC Viterbi School of Engineering

TRAINING PIPELINE

CLUSTERING - VALIDATION

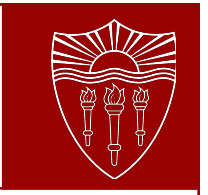


```python
@pipeline(name='hospital-readmission-training-pipeline')
def hospital_readmission_training_pipeline(
    training_dataset_path: str
):

    # Step 1: Split the dataset into train and validation sets
    data_split_step = split_train_val_data(
        input_data_path = training_dataset_path  # Use the parameter for training dataset
    )

    # Step 2: Initial training data preparation
    training_data_preparation = process_training_data(
        train_dataset = data_split_step.outputs['train_dataset']
    )

    # Step 3: Initial validation data preparation
    validation_data_preparation = process_validation_data(
        val_dataset = data_split_step.outputs['val_dataset'],
        scaler_model = training_data_preparation.outputs['scaler_model'],
        all_mean = training_data_preparation.outputs['all_mean']
    )

    # Step 4: Perform SMOTE oversampling on clustered training data
    oversampled_training_data = perform_SMOTE(
        input_df = training_data_preparation.outputs['processed_train_dataset']
    )

    # Step 5: Train XGBoost model
    trained_model = train_boost_model(
        training_dataset=oversampled_training_data.outputs['output_df']
    )

    # Step 6: Evaluate the model
    evaluate_model(
        test_dataset = validation_data_preparation.outputs['processed_val_dataset'],
        trained_model=trained_model.outputs['trained_model']
    )
```

USC Viterbi School of Engineering

TRAINING PIPELINE

PERFORM_SMOTE



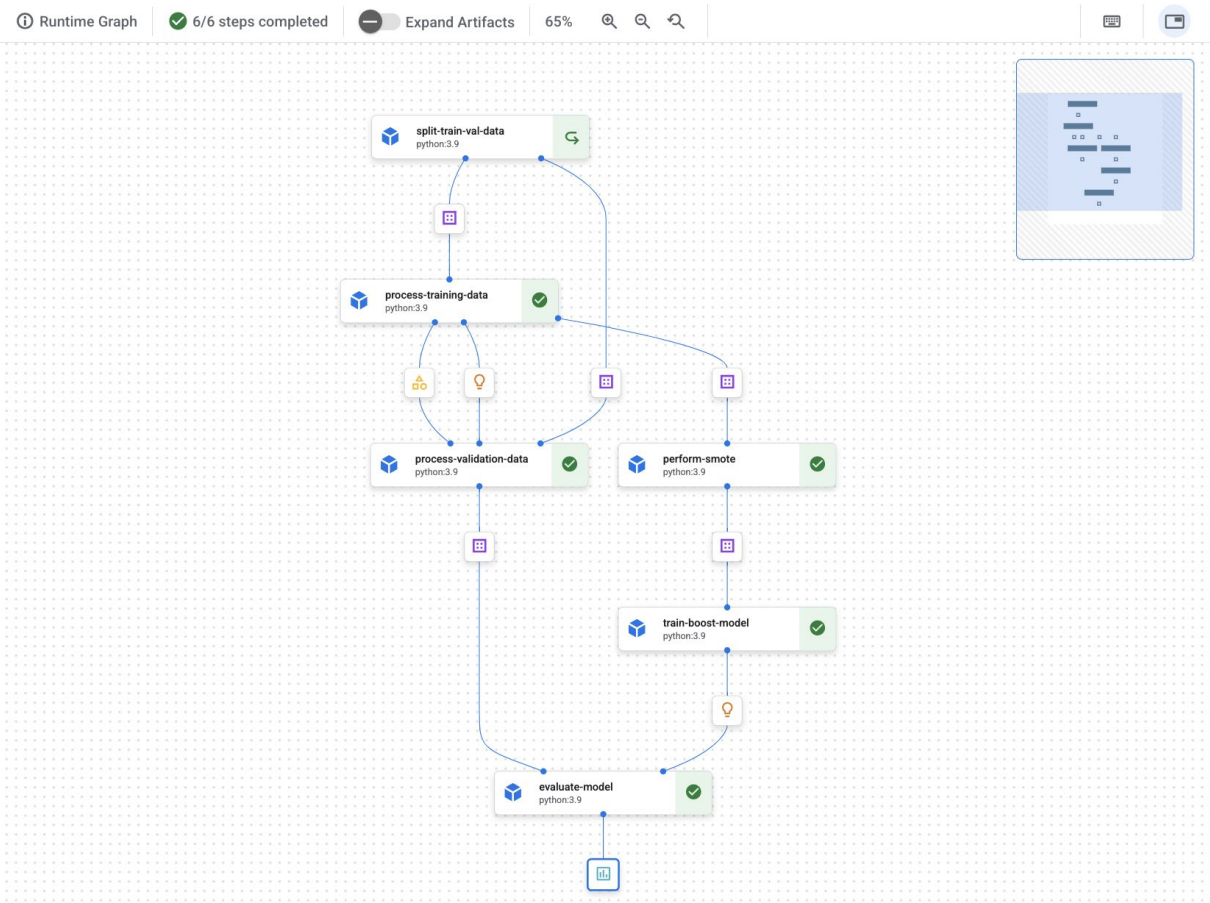
```
[53] compiler.Compiler().compile(
        pipeline_func = hospital_readmission_training_pipeline,
        package_path = 'hospital_readmission_training_pipeline.json'
    )

    pipeline_job = aiplatform.PipelineJob(
        display_name = 'hospital_readmission_training_pipeline_job',
        template_path = 'hospital_readmission_training_pipeline.json',
        pipeline_root = readmission_dataset_path,
        parameter_values = {
          'training_dataset_path': f'{readmission_dataset_path}/healthcare_readmissions_dataset_train.csv'
        }
    )

    pipeline_job.submit()
```

```
INFO:google.cloud.aiplatform.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created. Resource name: projects/1050269307600/locations/us-cent
INFO:google.cloud.aiplatform.pipeline_jobs:To use this PipelineJob in another session:
INFO:google.cloud.aiplatform.pipeline_jobs:pipeline_job = aiplatform.PipelineJob.get('projects/1050269307600/locations/
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/hospital-readmission-training-pipeline-
```

USC Viterbi School of Engineering

TRAINING PIPELINE

MODEL ASSESSMENT STATISTICS



Summary | Node info

Artifact Info

Open in ML Metadata

Name	metrics
Type	system.Metrics
URI	gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/evaluate-model_2791782240512638976/metrics

Metrics

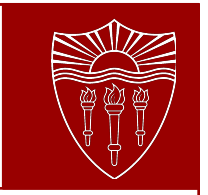
Scalar metrics produced by this step.

accuracy	0.8439054726368159
f1_score	0.7724652454422659
false_negatives	251
false_positives	0
true_negatives	1357
true_positives	0

USC Viterbi School of Engineering

INFERENCE PIPELINE

DEFINE INFERENCE PIPELINE



```
model_path = 'gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/train-boost-model_-6628903755016306688/trained_model'
scaler_path = 'gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/process-training-data_4900311291052163072/scaler_model'
mean_path = 'gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/process-training-data_4900311291052163072/all_mean'
```

```
import pandas as pd
imputed_mean_artifact_path = "gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/process-training-data_4900311291052163072/executor_output.json"

imputed_mean_dictionary = pd.read_json(imputed_mean_artifact_path).to_dict()
imputed_mean_dictionary
```


```
{'artifacts': {'all_mean': {'artifacts': [{'name': 'projects/1050269307600/locations/us-central1/metadataStores/default/artifacts/12978531633332825206',
     'uri': 'gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/process-training-data_4900311291052163072/all_mean',
     'metadata': {'value': 2.547122861586314,
      'Age': 51.19206842923795,
      'Height (m)': 1.701477449455676,
      'BMI': 26.233094867807154,
      'Weight (kg)': 77.09760497667186,
      'Adjusted Weight (kg)': 76.24570588140081,
      'Has Diabetes': 0.13141524105754201,
      'Has Hypertension': 0.173716951788491,
      'Number of Prior Visits': 3.058519236986744,
      'Medications Prescribed': 3.517814726840855,
      'Length of Stay': 2.547122861586314}}]},
  'processed_train_dataset': {'artifacts': [{'name': 'projects/1050269307600/locations/us-central1/metadataStores/default/artifacts/5013363833885419394',
     'uri': 'gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/process-training-data_4900311291052163072/processed_train_dataset',
     'metadata': {}}]},
  'scaler_model': {'artifacts': [{'name': 'projects/1050269307600/locations/us-central1/metadataStores/default/artifacts/7037471566483568088',
     'uri': 'gs://543final_readmission/1050269307600/hospital-readmission-training-pipeline-20250506201355/process-training-data_4900311291052163072/scaler_model',
     'metadata': {}}]}}}
```

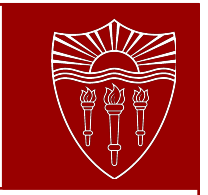

```
training_mean = imputed_mean_dictionary['artifacts']['all_mean']['artifacts'][0]['metadata']
training_mean = {k: v for k, v in training_mean.items() if k != "value"}
training_mean
```


```
{'Age': 51.19206842923795,
 'Height (m)': 1.701477449455676,
 'BMI': 26.233094867807154,
 'Weight (kg)': 77.09760497667186,
 'Adjusted Weight (kg)': 76.24570588140081,
 'Has Diabetes': 0.13141524105754201,
 'Has Hypertension': 0.173716951788491,
 'Number of Prior Visits': 3.058519236986744,
 'Medications Prescribed': 3.517814726840855,
 'Length of Stay': 2.547122861586314}
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

PERFORM INITIAL DATA PREPARATION



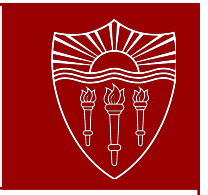
```python
from kfp.v2.dsl import component, Input, Output, Dataset, Model

@component(
    packages_to_install=["pandas", "scikit-learn", "joblib", "fsspec", "gcsfs"],
)
def process_prediction_data(
    prediction_dataset_path: str,
    processed_prediction_dataset: Output[Dataset],
    scaler_path: str,
    training_mean: dict,
):
    import pandas as pd
    import numpy as np
    import joblib
    from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
    import gcsfs
    import fsspec


    # Load training dataset
    df = pd.read_csv(prediction_dataset_path)

    # Separate features and target column
    ids = df['PatientID']
    X_test = df.drop(columns=['PatientID'])

    # Missing values
    numeric_cols = X_test.select_dtypes(include=['int64', 'float64']).columns
    categorical_cols = X_test.select_dtypes(include=['object', 'bool']).columns

    for col in numeric_cols:
      X_test[col] = X_test[col].fillna(training_mean.get(col))

    for col in categorical_cols:
      X_test[col] = X_test[col].fillna('missing')

    # Encode categorical variables
    categorical_cols = X_test.select_dtypes(include=['object', 'bool']).columns
    label_encoders = {}
    for col in categorical_cols:
        label_encoders[col] = LabelEncoder()
        X_test[col] = label_encoders[col].fit_transform(X_test[col])
```

```python
# Log-transform skewed features
X_test['Length of Stay_log'] = np.log1p(X_test['Length of Stay'])
X_test['Medications Prescribed_log'] = np.log1p(X_test['Medications Prescribed'])
X_test['Number of Prior Visits_log'] = np.log1p(X_test['Number of Prior Visits'])

# Load scaler from GCS
fs = gcsfs.GCSFileSystem()
with fs.open(scaler_path, 'rb') as f:
    scaler = joblib.load(f)

# Transform using the trained scaler
scaled_data = scaler.transform(X_test)

scaled_df = pd.DataFrame(scaled_data, columns=X_test.columns)
scaled_df.to_csv(processed_prediction_dataset.path, index=False)
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

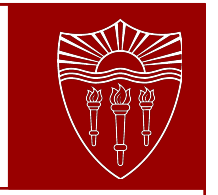
PERFORM INITIAL DATA PREPARATION



```python
from kfp.v2.dsl import component, Input, Output, Dataset, Model

@component(packages_to_install=["pandas", "numpy", "scikit-learn", "joblib", "fsspec", "gcsfs", "xgboost"])

def perform_predictions(
    dataset_for_prediction: InputPath('Dataset'),
    predictions_path: OutputPath('Dataset'),
    prediction_dataset_path: str,
    model_path: str
):

    import pandas as pd
    import joblib
    import gcsfs
    import xgboost


    # Create a GCS file system object and load the model
    fs = gcsfs.GCSFileSystem()
    with fs.open(model_path, 'rb') as f:
      trained_model = joblib.load(f)

    # Load the test dataset
    pred_df = pd.read_csv(dataset_for_prediction)

    # Make predictions
    y_pred = trained_model.predict(pred_df)
    pred_df['readmission_prediction'] = y_pred

    df = pd.read_csv(prediction_dataset_path)
    ids = df['PatientID']
    pred_df['PatientID'] = df['PatientID']


    pred_df = pred_df[['PatientID', 'readmission_prediction']]

    # Save the predictions
    pred_df.to_csv(predictions_path, index=False)
```

INFERENCE PIPELINE

IMPUTE AGE



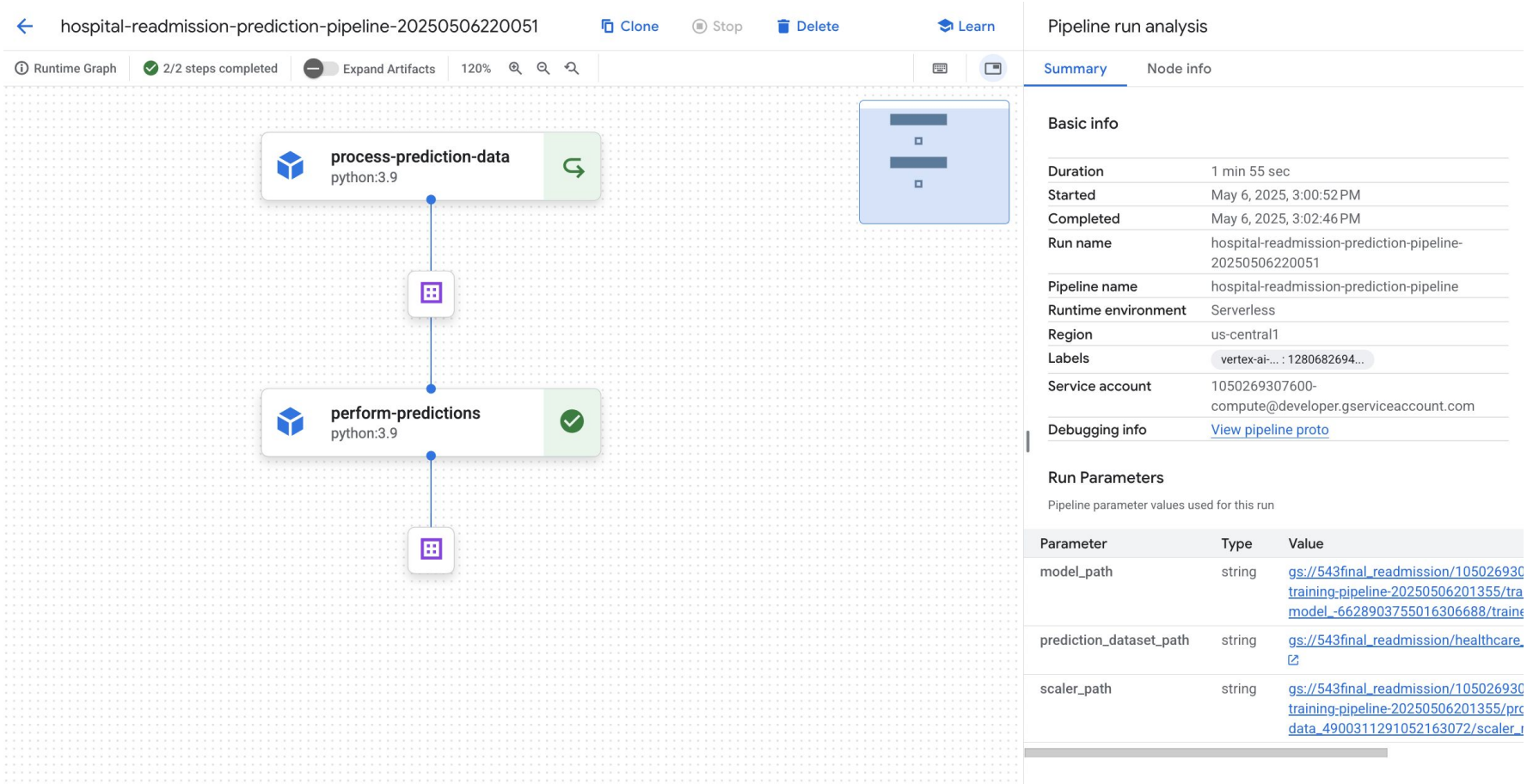
```python
from kfp.dsl import pipeline

@pipeline(name="Hospital Readmission Prediction Pipeline")
def hospital_readmission_prediction_pipeline(
    prediction_dataset_path: str,
    model_path: str,
    scaler_path: str
):
    process_step = process_prediction_data(
        prediction_dataset_path=prediction_dataset_path,
        scaler_path=scaler_path,
        training_mean = training_mean
    )

    perform_predictions(
        dataset_for_prediction=process_step.outputs["processed_prediction_dataset"],
        model_path=model_path,
        prediction_dataset_path=prediction_dataset_path
    )
```

```python
compiler.Compiler().compile(
    pipeline_func = hospital_readmission_prediction_pipeline,
    package_path = 'hospital_readmission_inference_pipeline.json'
)

pipeline_job = aiplatform.PipelineJob(
    display_name='hospital_readmission_inference_pipeline',
    template_path='hospital_readmission_inference_pipeline.json',
    pipeline_root = readmission_dataset_path,
    parameter_values={
        'prediction_dataset_path': f'{readmission_dataset_path}/healthcare_readmissions_dataset_test.csv',
        'model_path' : model_path,
        'scaler_path' : scaler_path
    },
    enable_caching=True
)

pipeline_job.submit()
```

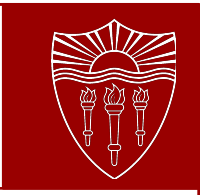
USC Viterbi School of Engineering

INFERENCE PIPELINE

PIPELINE DESIGN



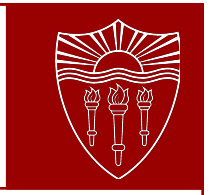
SUMMARY DISCUSSION



- Feature engineering decisions that improved the model
  - SMOTE is critical, increasing f1 by 5%
- Feature selection can be done based on the feature importance
  - Variables with low feature importance can be removed
  - Low feature importance variables: Ethnicity, Type of Treatment, Adjusted Weight, Hospital ID, Height, Gender (descending sorted)
- XGBoost perform best so far
  - Next steps would be to try manual more complicated ensemble models
  - Or put the AutoML model into the inference pipeline to make the prediction.

SUMMARY DISCUSSION

Model Experiment 1



- New model with feature selection was tried, the training F1 score increase, but the testing f1 score doesn't show significant improvement.
- Model performance on validation set:

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1357
1	0.76	0.81	0.79	251
accuracy			0.93	1608
macro avg	0.86	0.88	0.87	1608
weighted avg	0.93	0.93	0.93	1608

- Model performance on testing set

1) F1-Score (76.01/100)

```
Test Failed: 76.01 != 100 : The F1-Score is 0.7601156069364162.
```