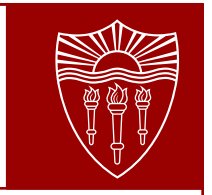
USC Viterbi
School of Engineering

FINAL REPORT

YIWEN MEI

USC Viterbi
School of Engineering

FINAL REPORT
OUTLINE

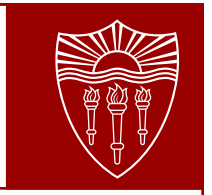


- Exploratory Data Analysis Report
- Data Preparation Plan
- Model Pipeline
- Summary Discussion

USC Viterbi
School of Engineering

EDA REPORT

OUTLINE

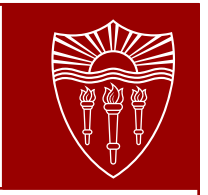


- Business objective
- Dataset summary
- Data quality summary
- Univariate analysis
- Bivariate analysis

USC Viterbi School of Engineering

EDA REPORT

BUSINESS OBJECTIVE



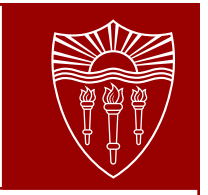
- Develop a model to predict whether patients are at high risk of developing coronary heart disease (CHD) within the next ten years using a healthcare dataset of patient information

- Impact:
- This allows for targeted interventions and personalized treatment plans to prevent or delay the onset of CHD
- Health organizations can design more effective prevention campaigns, focusing on lifestyle changes, dietary guidance, and regular health screenings

USC Viterbi School of Engineering

EDA REPORT

DATASET SUMMARY



- "Final Project Dataset.csv" – dataset with 19 variables and 3,816 observations
- Response variable:

Name	Description
TenYearCHD	Whether the participant developed coronary heart disease (CHD) within 10 years of the examination (yes or no)

USC Viterbi School of Engineering

DATASET SUMMARY

DEMOGRAPHIC VARIABLES

Demographic variables describe the profile of each patient in terms of age, gender, income and education (age: numerical, male: categorical, education: categorical, income: numerical)

Name	Description
Age	age of the participant at the time of examination
Male	gender of the participant (male =1, female = 0)
Education	Educational level of the patient (1 = less than high school, 2 = completed high school or equivalent, 3 = some college, 4= completed college or higher)
Income	Income of the patient

USC Viterbi School of Engineering

DATASET SUMMARY

OTHER CATEGORICAL VARIABLES

Summarized key health indicators such as smoking status, use of blood pressure medication, history of stroke, hypertension, and diabetes, each of which can significantly influence a patient's health status

Name	Description
Current Smoker	whether the participant is currently a smoker (yes or no)
BP Meds	whether the participant is taking blood pressure medication (yes or no)
Prevalent Stroke	whether the participant has a history of stroke (yes or no)
Prevalent Hyp	whether the participant has a history of hypertension (yes or no)
Diabetes	whether the participant has diabetes (yes or no)

USC Viterbi School of Engineering

DATASET SUMMARY

OTHER NUMERICAL VARIABLES

Numerical measures on a patient's health, including daily cigarette consumption, cholesterol levels, blood pressure measurements, body mass index, heart rate, blood glucose levels, and hemoglobin A1c percentage

Name	Description
Cigarettes per Day	the average number of cigarettes smoked per day by current smokers
Total Chol	total cholesterol level in milligrams per deciliter
Sys BP	systolic blood pressure in millimeters of mercury
Dia BP	diastolic blood pressure in millimeters of mercury
BMI	body mass index in kilograms per square meter
Heart Rate	resting heart rate in beats per minute
Glucose	Blood glucose level in milligrams per deciliter
A1c	Hemoglobin A1c (%)

EDA REPORT

DATA QUALITY SUMMARY

Training dataset contains 19 variables and 3052 observations (80-20 split, random_state = 42)

Missing values:

```
1 train_df = pd.read_csv('/content/drive/My Drive/train_data.csv')
2 train_df.isnull().sum()
```

```
patientID            0
male                 0
age                  0
education           72
currentSmoker        0
cigsPerDay        1549
BPMeds              39
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             38
sysBP                0
diaBP                0
BMI                 15
heartRate            1
glucose            292
TenYearCHD           0
a1c                292
income               0
dtype: int64
```

Eight variables have missing values:

Categorical (Imputed with mode):

- education
- BPMeds

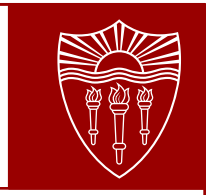
Numerical (will show how to deal with missing values in Page 23):

- cigsPerDay
- totChol
- BMI
- heartRate
- glucose
- a1c (will be dropped because it is highly correlated with glucose)

USC Viterbi School of Engineering

EDA REPORT

DATA QUALITY SUMMARY



Data types:

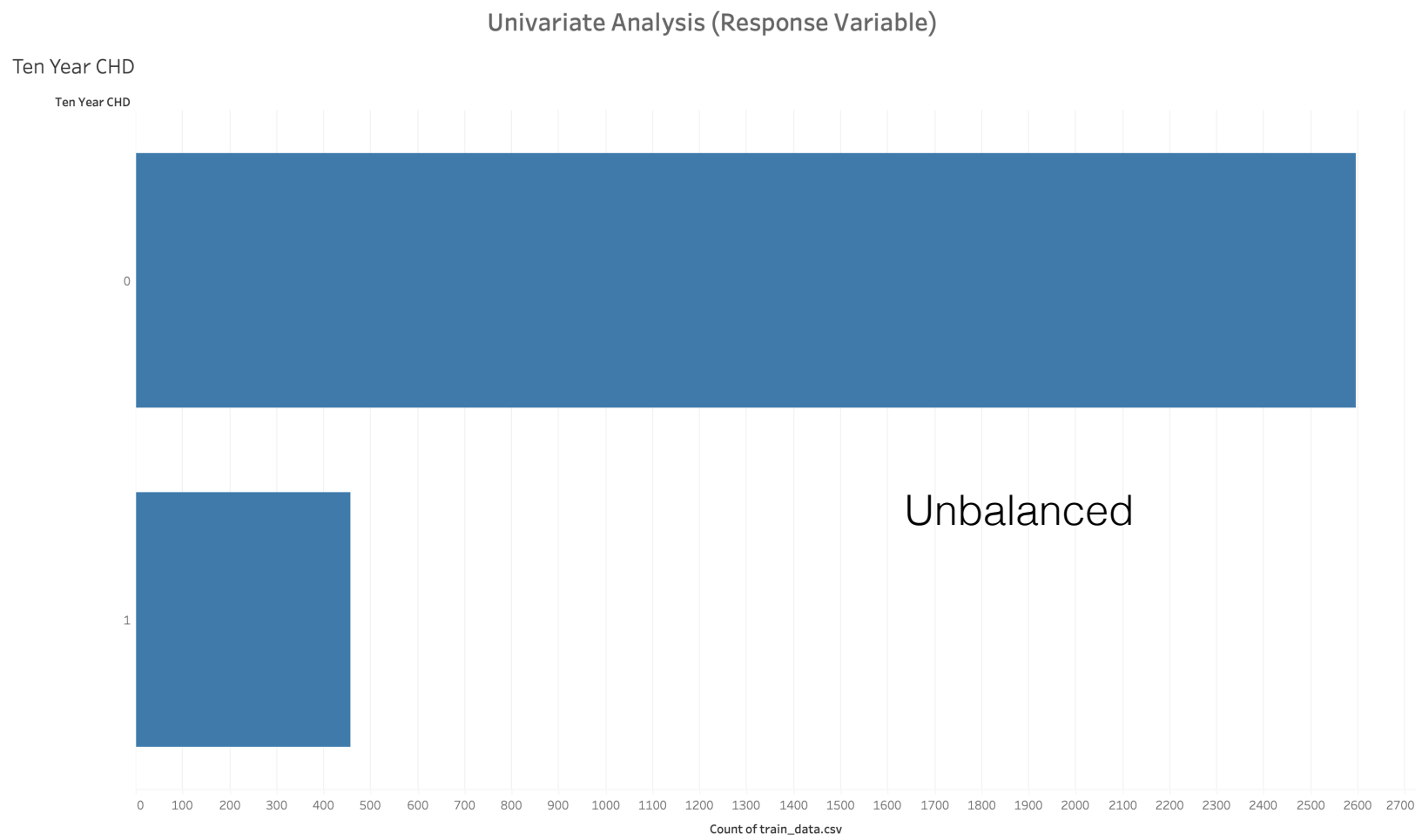
```
1 train_df.dtypes
```

```
patientID            int64
male                 int64
age                  int64
education          float64
currentSmoker        int64
cigsPerDay         float64
BPMeds             float64
prevalentStroke      int64
prevalentHyp         int64
diabetes             int64
totChol            float64
sysBP              float64
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
TenYearCHD           int64
a1c                float64
income               int64
dtype: object
```

USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

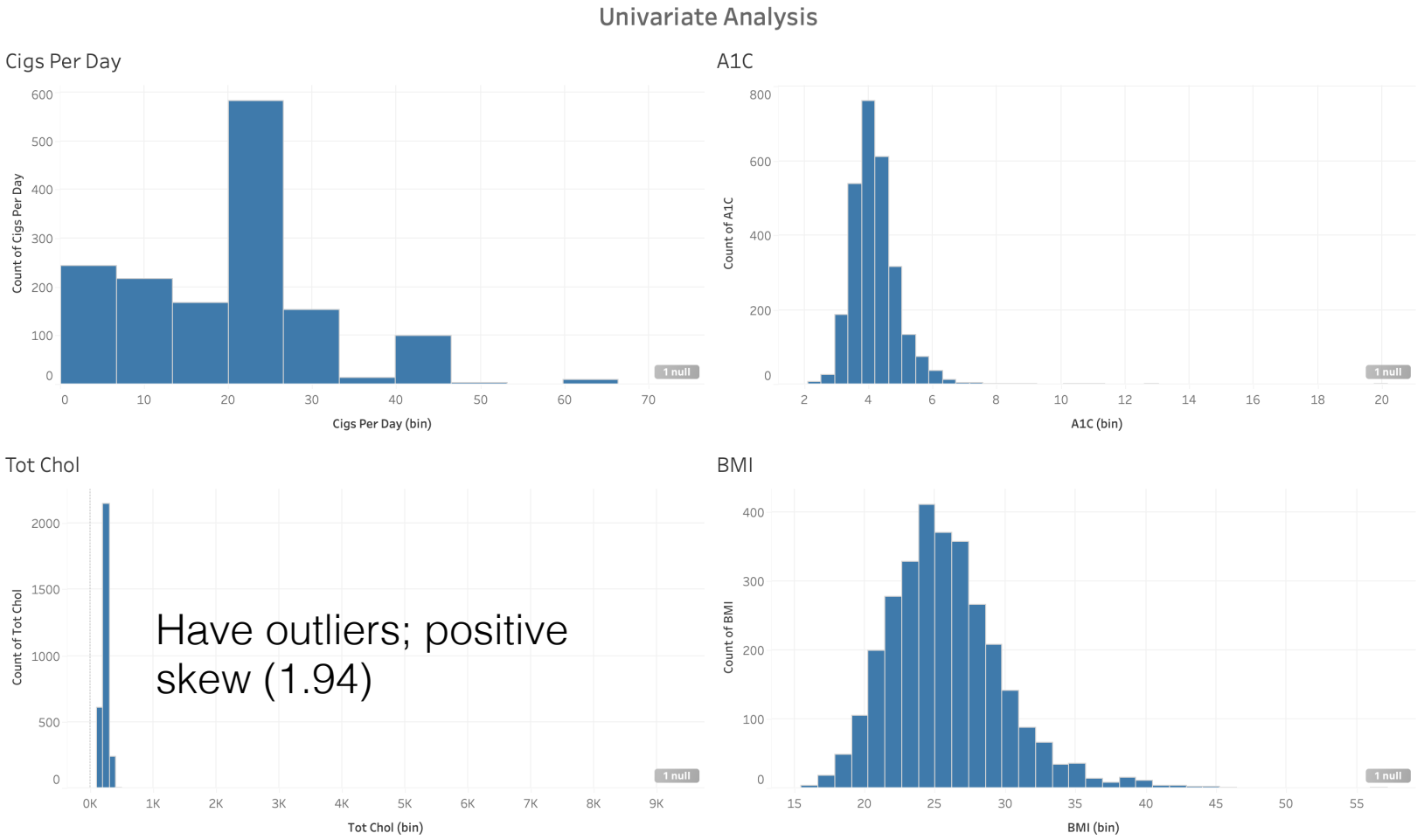
RESPONSE VARIABLES



USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

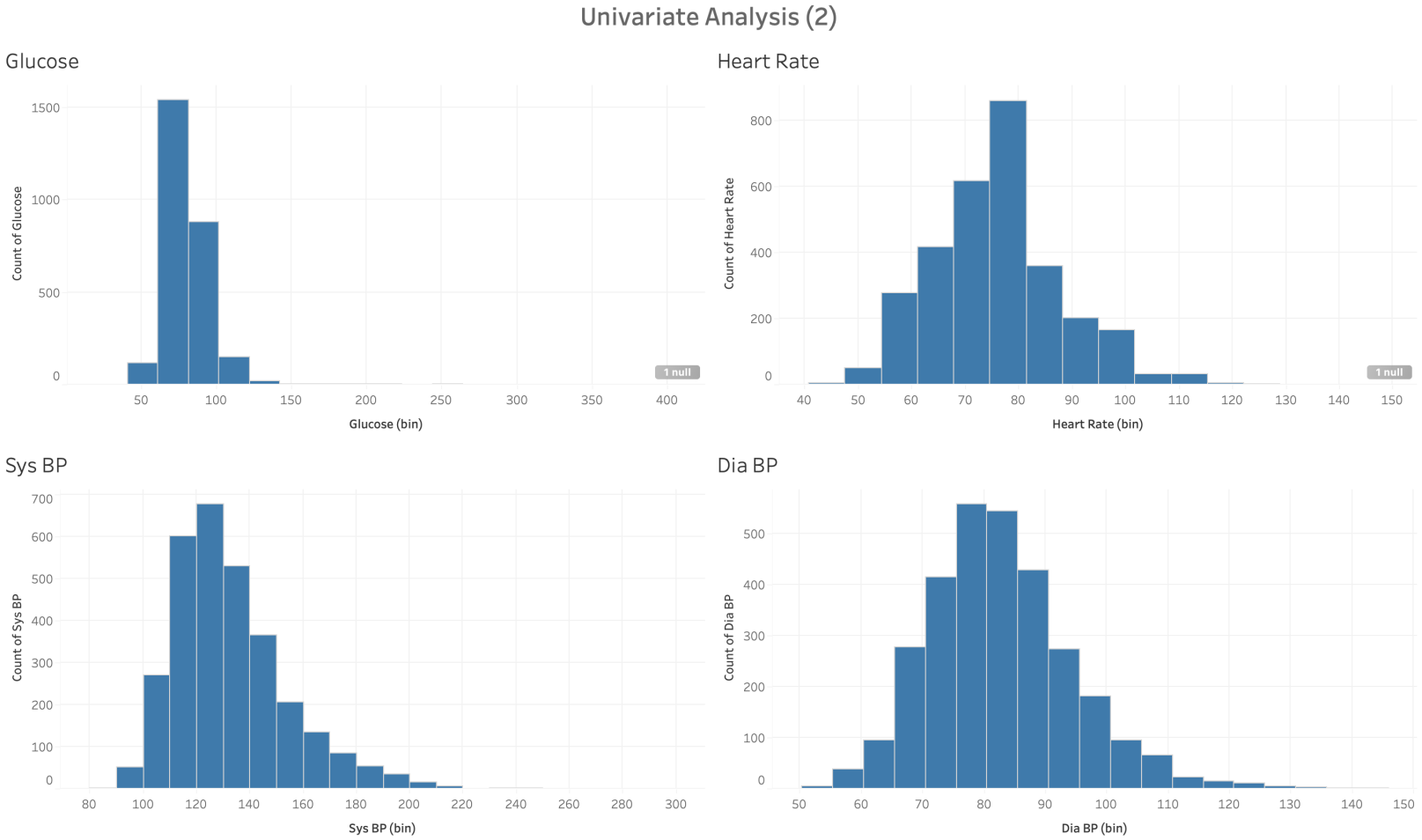
NUMERICAL VARIABLES



USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

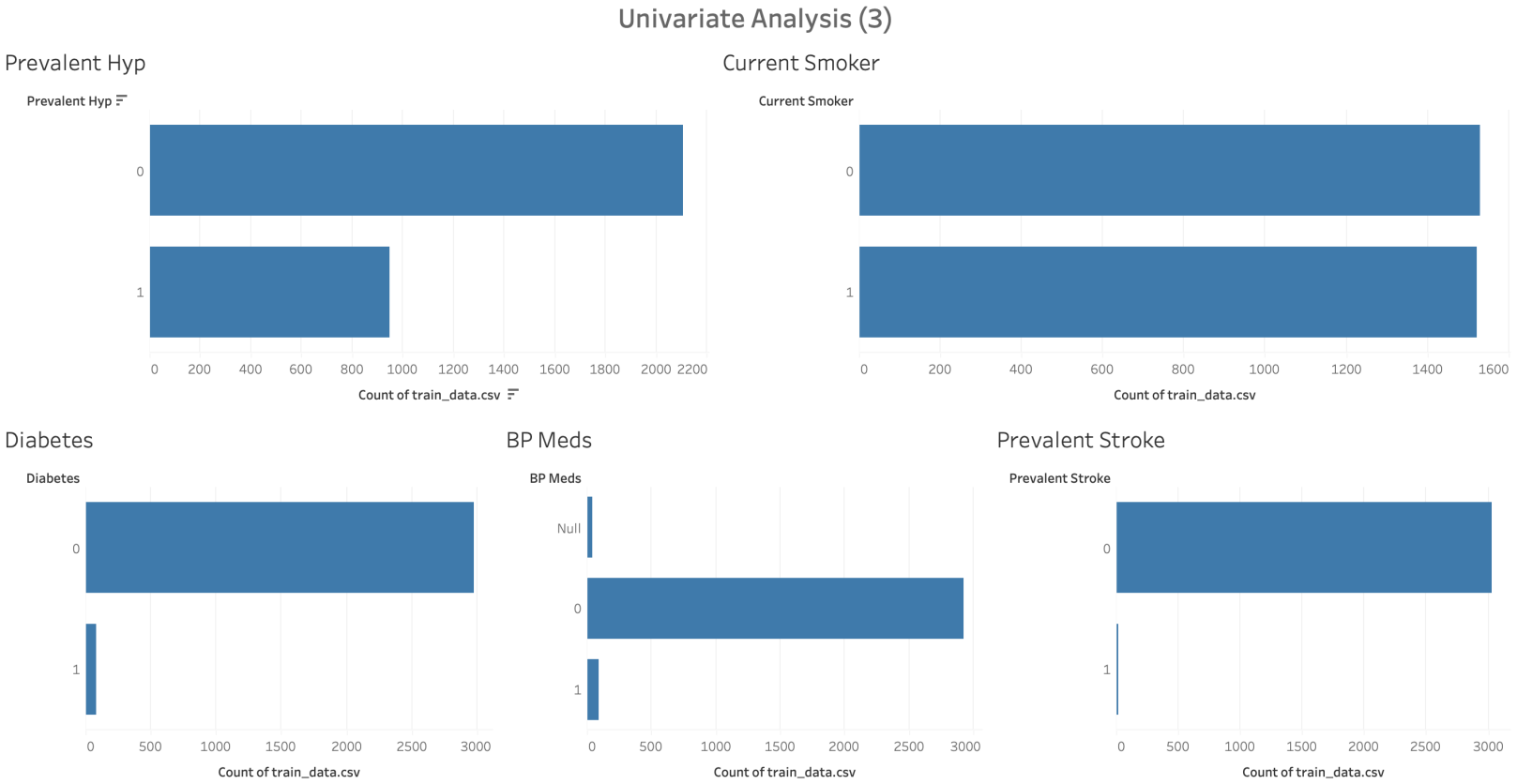
NUMERICAL VARIABLES



USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

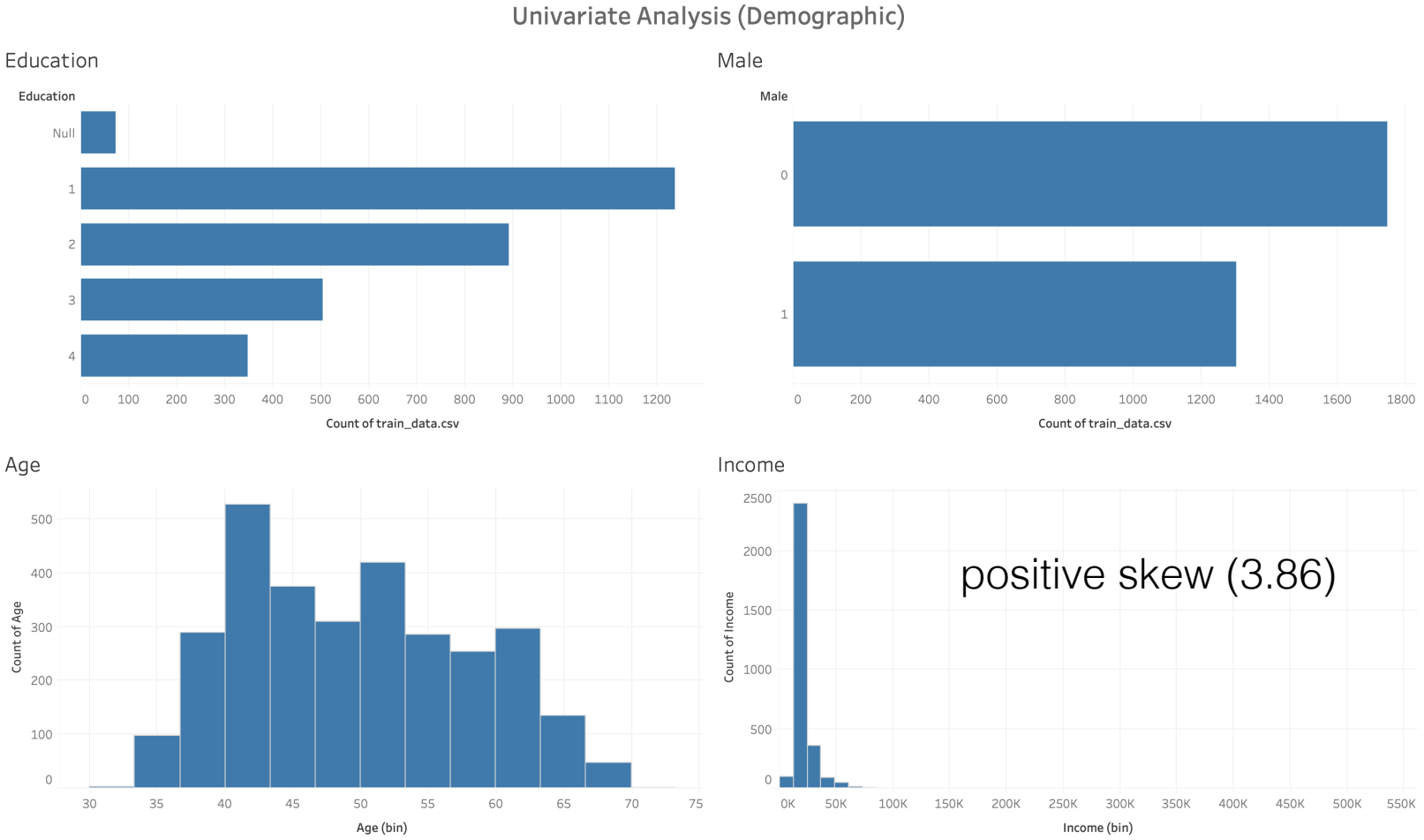
CATEGORICAL VARIABLES



USC Viterbi School of Engineering

EDA REPORT – UNIVARIATE ANALYSIS

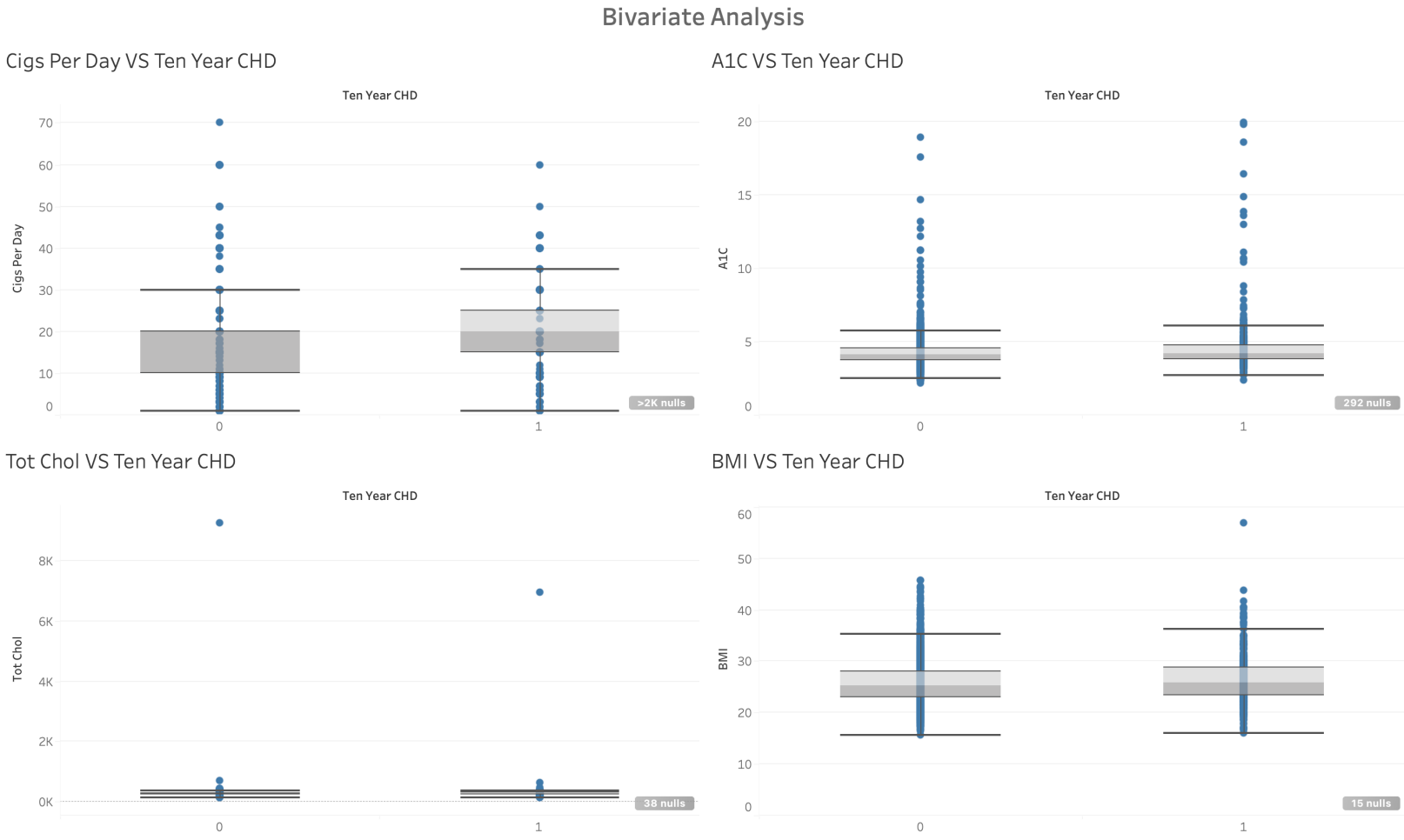
DEMOGRAPHIC VARIABLES



USC Viterbi School of Engineering

BIVARIATE ANALYSIS

NUMERICAL VARIABLES VS RESPONSE



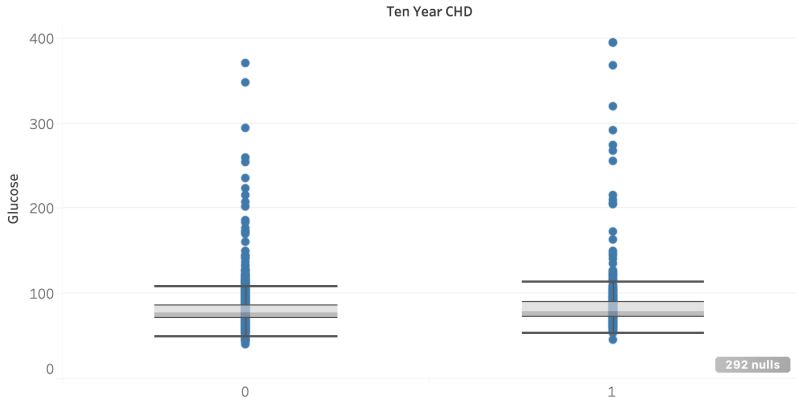
USC Viterbi School of Engineering

BIVARIATE ANALYSIS

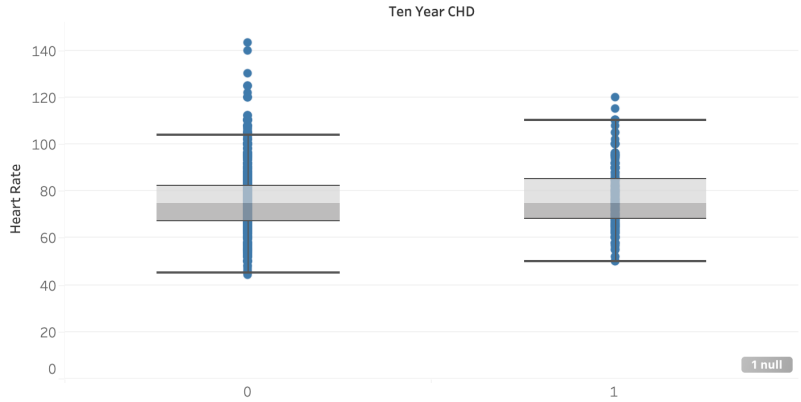
NUMERICAL VARIABLES VS RESPONSE

Bivariate Analysis (2)

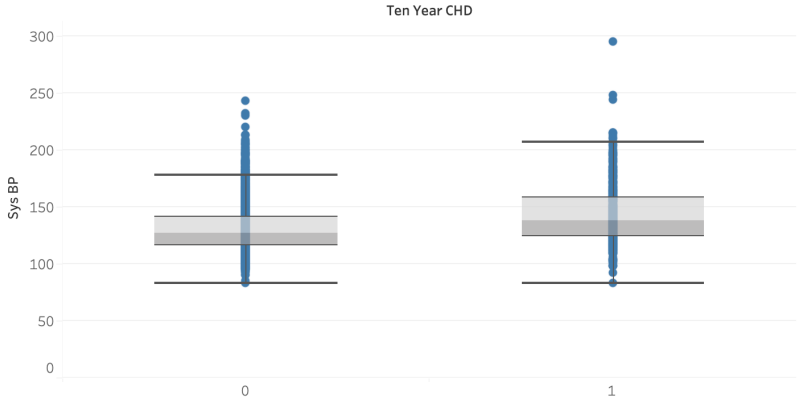
Glucose VS Ten Year CHD



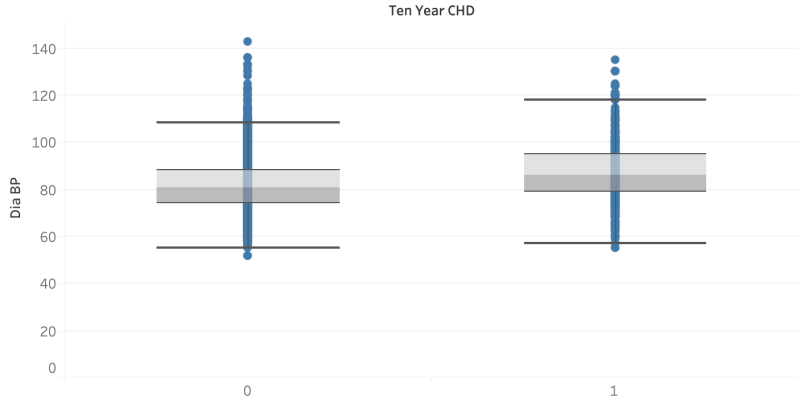
Heart Rate VS Ten Year CHD



Sys BP VS Ten Year CHD



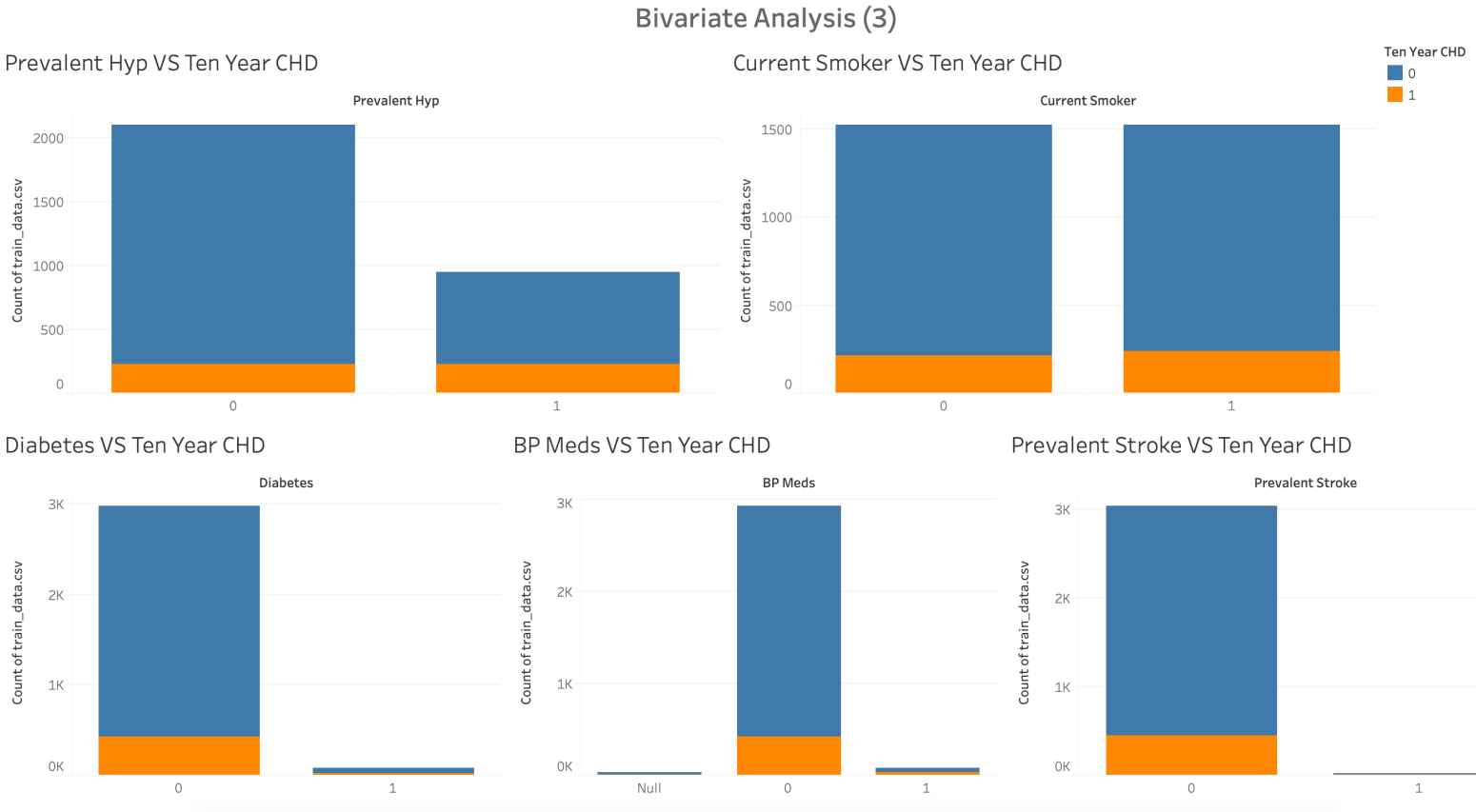
Dia BP VS Ten Year CHD



USC Viterbi School of Engineering

BIVARIATE ANALYSIS

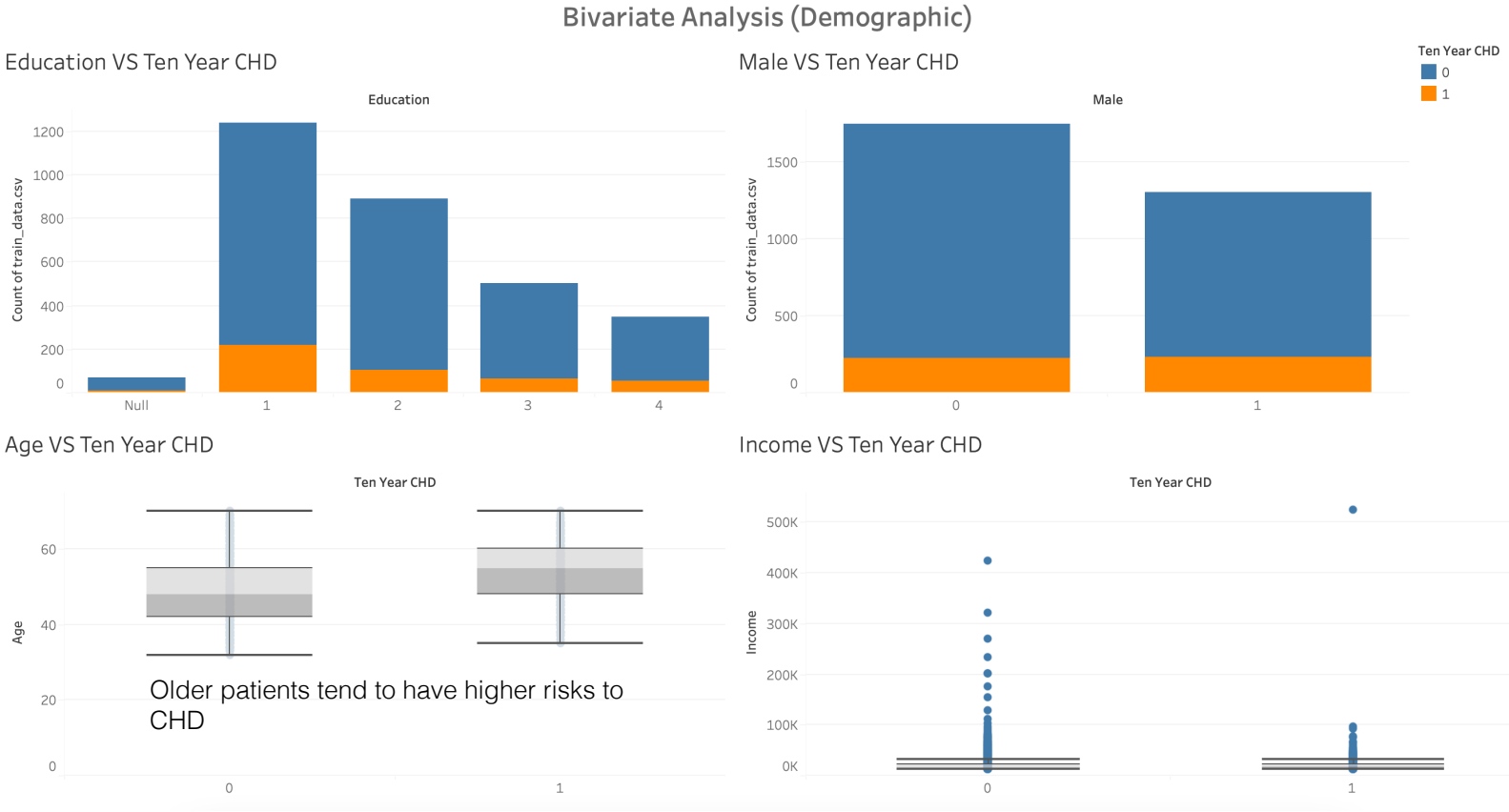
CATEGORICAL VARIABLES VS RESPONSE



USC Viterbi School of Engineering

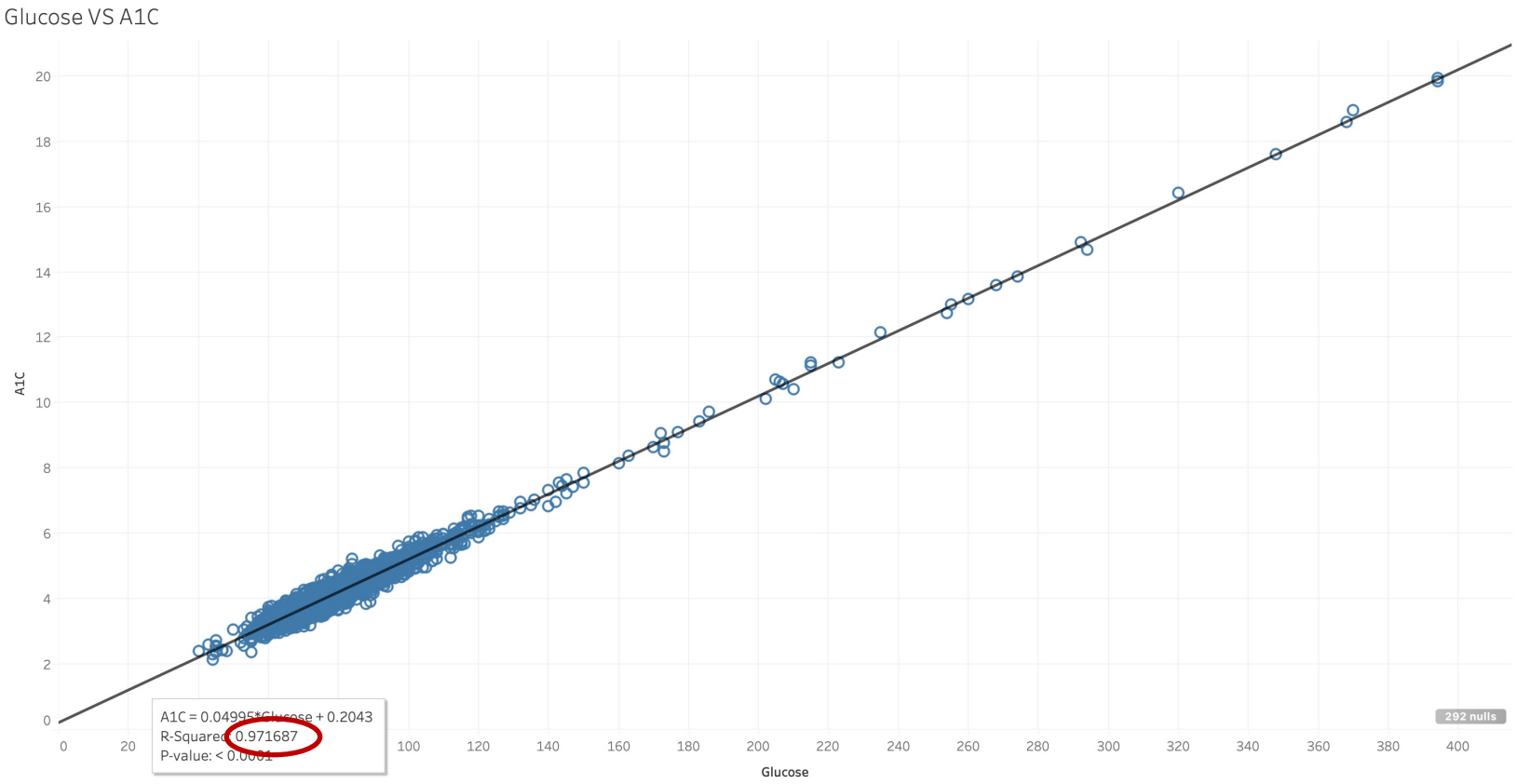
BIVARIATE ANALYSIS

DEMOGRAPHIC VARIABLES VS RESPONSE



BIVARIATE ANALYSIS
OTHER PAIRS OF VARIABLES

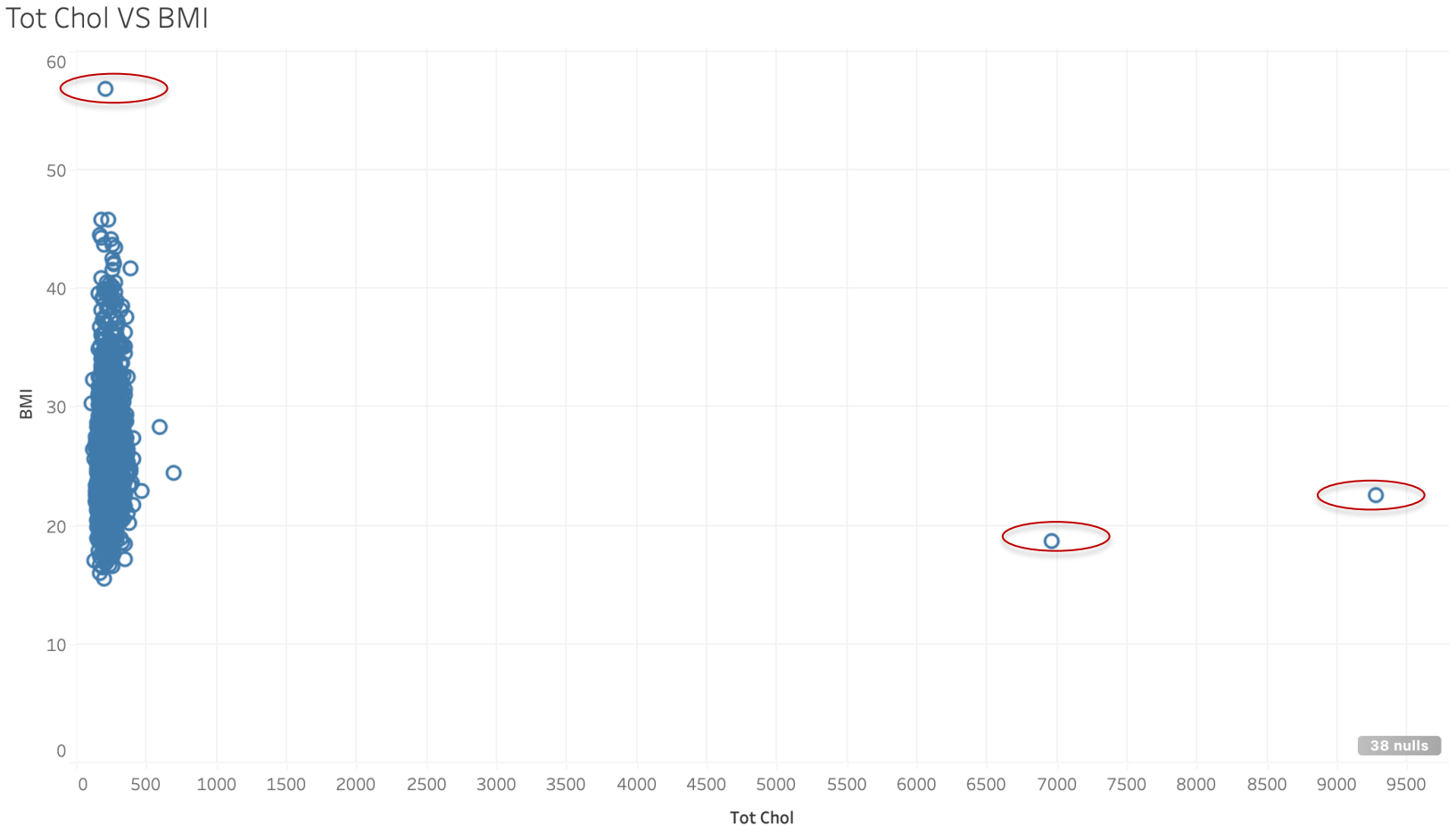
glucose VS a1c



glucose and a1c are highly correlated. We will drop a1c.

USC Viterbi School of Engineering

BIVARIATE ANALYSIS
OTHER PAIRS OF VARIABLES

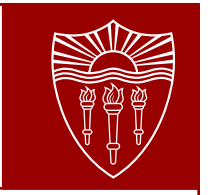


We observe outliers for totChol and BMI. Since the majority of BMI observations is under 50 and totChol is under 700, We will “clip” BMI at 50 and totChol at 700.

USC Viterbi School of Engineering

DATA PREPARATION PLAN

OVERVIEW



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

DATA PREPARATION PLAN

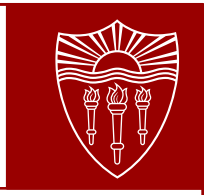
DATA QUALITY ISSUES AND ACTIONS

Missing variables

- Categorical (Imputed with mode):
  - education
  - BPMeds
- Numerical:
  - totChol: imputed with median
  - BMI: imputed with median
  - heartRate: imputed with median
  - glucose: imputed with median
  - a1c (will be dropped because it is highly correlated with glucose)
  - cigsPerDay: first imputed with median, then make some modifications, since it is impossible for non current smokers to have positive cigsPerDay. Therefore, if currentSmoker = 0, we change cigsPerDay from imputed median to 0.

DATA PREPARATION PLAN

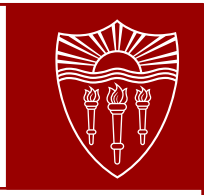
FEATURE ENGINEERING DECISIONS



- Log-transform totChol (after “clip”) and income due to strong positive skew

USC Viterbi School of Engineering

DATA PREPARATION PLAN
DATASET PARTITIONING DECISIONS

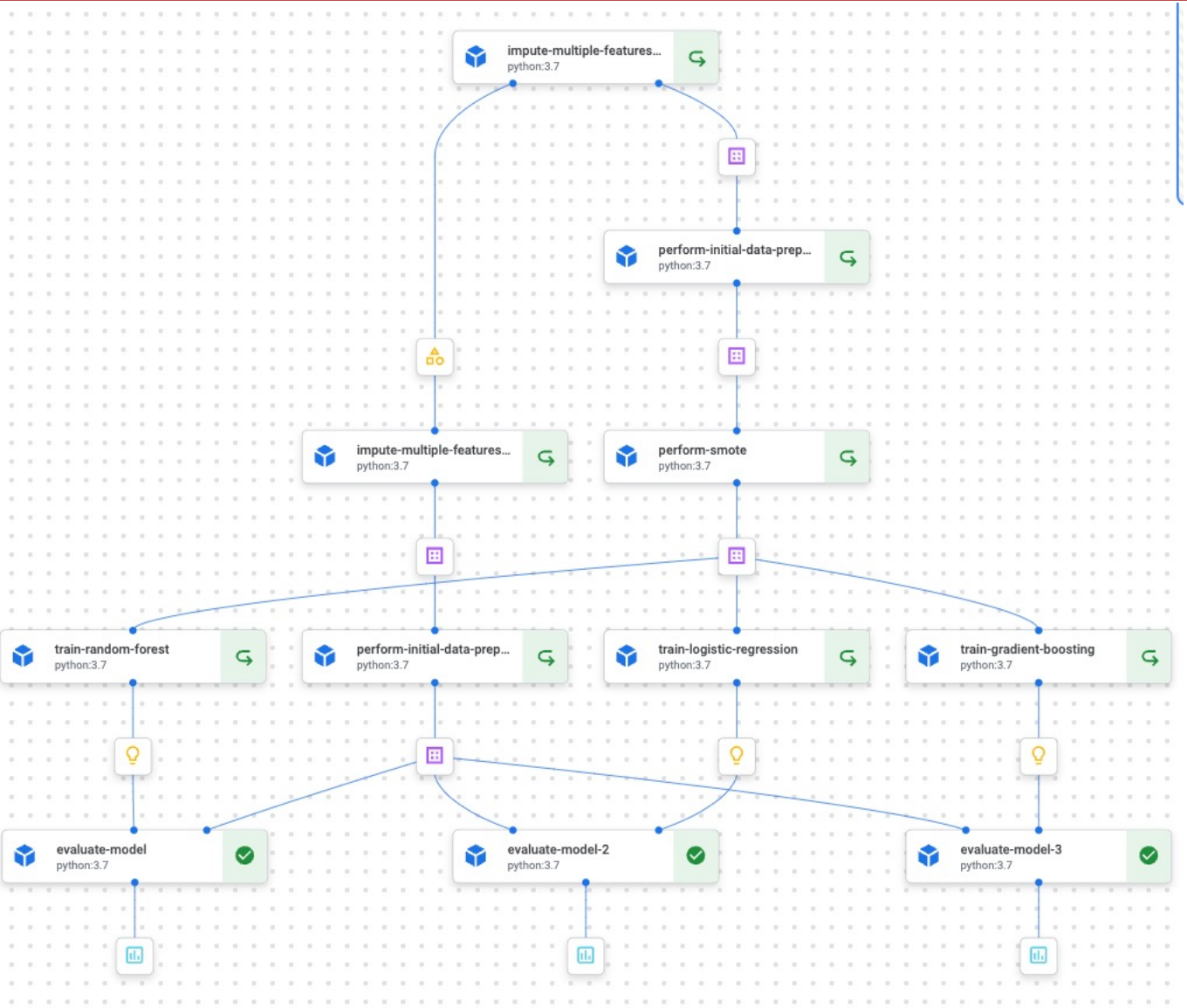


- Perform SMOTE oversampling due to unbalanced dataset

USC Viterbi School of Engineering

MODEL PIPELINE

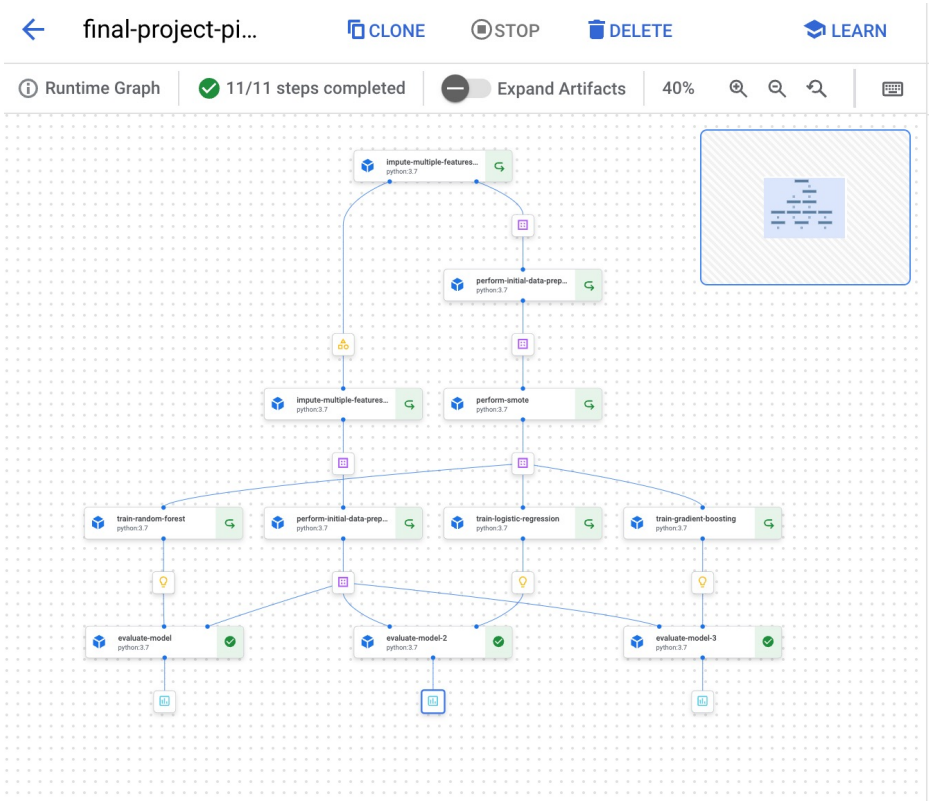
OVERALL PIPELINE



USC Viterbi School of Engineering

MODEL PIPELINE

MODEL EVALUATION RESULTS – LOGISTIC REGRESSION



Pipeline run analysis

SUMMARY | NODE INFO

Artifact Info

OPEN IN ML METADATA

Name	metrics
Type	system.Metrics
URI	gs://final-bucket-2/795244162107/final-project-pipeline-20240502235419/evaluate-model-2_4525461835540856832/metrics

Metrics

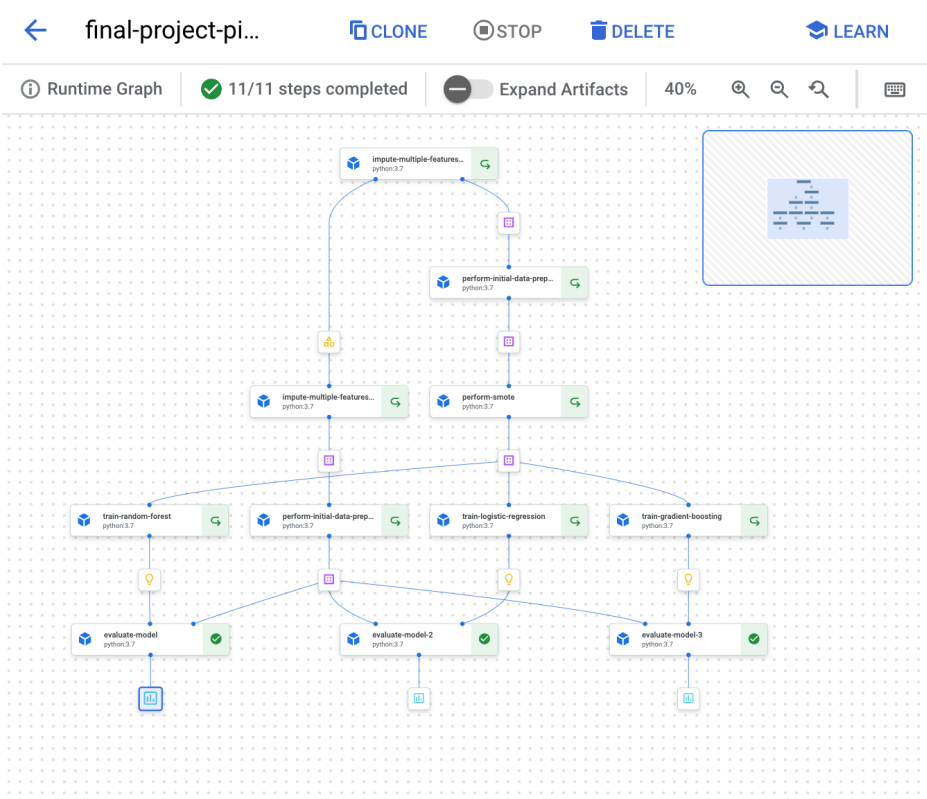
Scalar metrics produced by this step.

AUC_PR	0.6069787038455645
accuracy	0.6544502617801047
f1_score	0.6944877710089439

USC Viterbi School of Engineering

MODEL PIPELINE

MODEL EVALUATION RESULTS – RANDOM FOREST



Pipeline run analysis

SUMMARY | NODE INFO

Artifact Info

OPEN IN ML METADATA

Name	metrics
Type	system.Metrics
URI	gs://final-bucket-2/795244162107/final-project-pipeline-20240502235419/evaluate-model_-2392067192100225024/metrics

Metrics

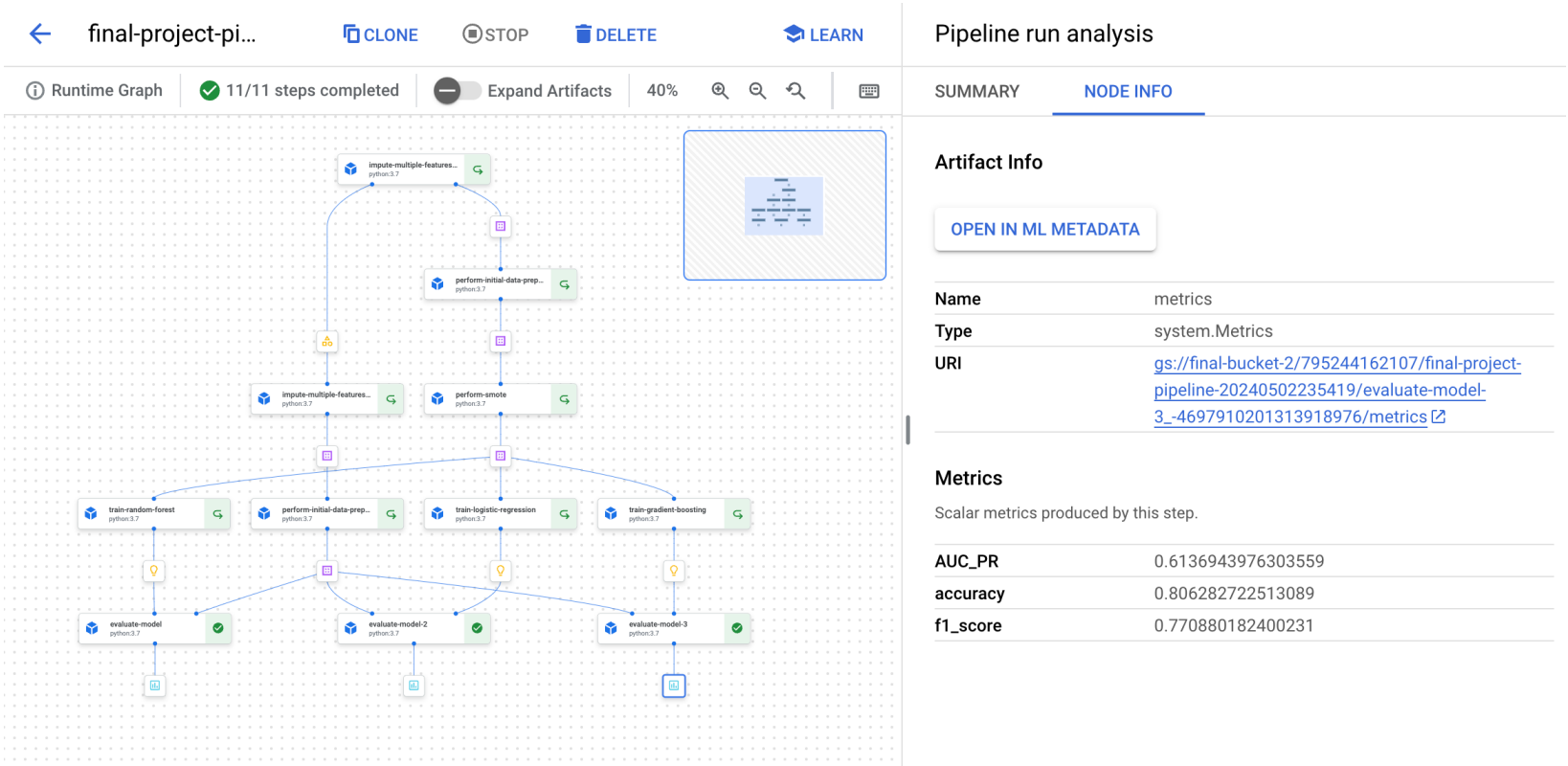
Scalar metrics produced by this step.

AUC_PR	0.6612404882283847
accuracy	0.7133507853403142
f1_score	0.7353675482843073

MODEL PIPELINE

MODEL EVALUATION RESULTS – GRADIENT BOOSTING

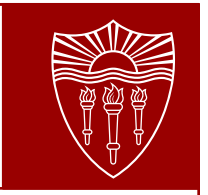
- Gradient Boosting has the highest f1_score among these three models, so we only use train-gradient-boosting model in the inference pipeline.



USC Viterbi School of Engineering

MODEL PIPELINE

PIPELINE DEFINITION CODE

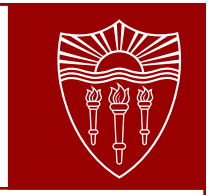


```
[44] @pipeline(name='final-project-pipeline')
     def final_project_pipeline(training_dataset_path: str, validation_dataset_path: str):

         # Perform imputation before data preparation
         training_imputation = impute_multiple_features_training(training_dataset_path=training_dataset_path)
         validation_imputation = impute_multiple_features_validation(validation_dataset_path=validation_dataset_path,
                                                                     feature_medians=training_imputation.outputs['feature_medians'])

         # Data preparation using the output from the imputation step
         training_data_preparation = perform_initial_data_preparation(input_dataset_path=training_imputation.outputs['imputed_dataset_path'])
         validation_data_preparation = perform_initial_data_preparation(input_dataset_path=validation_imputation.outputs['imputed_dataset_path'])

         # Perform SMOTE oversampling on the training partition
         oversampled_training_data = perform_SMOTE(input_df_path=training_data_preparation.outputs['output_dataset_path'])

         # Training and evaluation using the prepared datasets
         trained_random_forest = train_random_forest(training_dataset_path=oversampled_training_data.outputs['output_df_path'])
         evaluate_model(test_dataset_path=validation_data_preparation.outputs['output_dataset_path'],
                        model=trained_random_forest.outputs['trained_model_artifact'])

         trained_logistic_regression = train_logistic_regression(training_dataset_path=oversampled_training_data.outputs['output_df_path'])
         evaluate_model(test_dataset_path=validation_data_preparation.outputs['output_dataset_path'],
                        model=trained_logistic_regression.outputs['trained_model_artifact'])

         trained_gradient_boosting = train_gradient_boosting(training_dataset_path=oversampled_training_data.outputs['output_df_path'])
         evaluate_model(test_dataset_path=validation_data_preparation.outputs['output_dataset_path'],
                        model=trained_gradient_boosting.outputs['trained_model_artifact'])
```

USC Viterbi School of Engineering

COMPONENT DEFINITION

IMPUTE_MULTIPLE_FEATURES_TRAINING

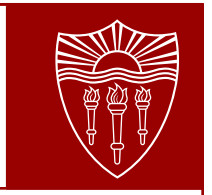


```python
from kfp.v2.dsl import Artifact
from kfp.v2.dsl import Input
from kfp.v2.dsl import Model

@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
def impute_multiple_features_training(training_dataset_path: str,
                                      imputed_dataset_path: OutputPath('Dataset'),
                                      feature_medians: Output[Artifact]):
    import pandas as pd

    df = pd.read_csv(training_dataset_path)
    features_to_impute = ['cigsPerDay', 'totChol', 'glucose', 'BMI', 'heartRate']
    medians = {}
    for feature in features_to_impute:
        median_value = df[feature].median()
        df[feature].fillna(median_value, inplace=True)
        medians[feature] = median_value

    categorical_features_to_impute = ['education', 'BPMeds']
    modes = {}
    for feature in categorical_features_to_impute:
        mode_value = df[feature].mode()[0]
        df[feature].fillna(mode_value, inplace=True)
        modes[feature] = mode_value

    feature_medians.metadata['medians'] = medians
    feature_medians.metadata['modes'] = modes

    df.to_csv(imputed_dataset_path, index=False)
```

USC Viterbi School of Engineering

COMPONENT DEFINITION

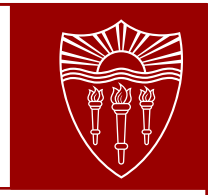
IMPUTE_MULTIPLE_FEATURES_VALIDATION



```
[31] @component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
     def impute_multiple_features_validation(validation_dataset_path: str,
                                             imputed_dataset_path: OutputPath('Dataset'),
                                             feature_medians: Input[Artifact]):
         import pandas as pd

         df = pd.read_csv(validation_dataset_path)
         medians = feature_medians.metadata['medians']
         modes = feature_medians.metadata['modes']

         for feature, median_value in medians.items():
             df[feature].fillna(median_value, inplace=True)

         for feature, mode_value in modes.items():
             df[feature].fillna(mode_value, inplace=True)

         df.to_csv(imputed_dataset_path, index=False)
```

COMPONENT DEFINITION

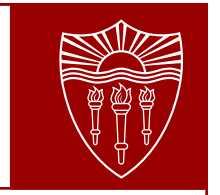
PERFORM_INITIAL_DATA_PREPARATION



```python
@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
def perform_initial_data_preparation(input_dataset_path: InputPath('Dataset'),
                                     output_dataset_path: OutputPath(Dataset)):
    import pandas as pd
    import numpy as np

    df = pd.read_csv(input_dataset_path)
    df = df.drop(columns=["a1c"])
    df["totChol"] = df["totChol"].clip(upper=700)
    df["BMI"] = df["BMI"].clip(upper=50)
    df["totChol"] = np.log(df["totChol"]+1)
    df["income"] = np.log(df["income"]+1)
    df.loc[df['currentSmoker'] == 0, 'cigsPerDay'] = 0
    df = pd.get_dummies(df, drop_first=True)
    df.to_csv(output_dataset_path, index=False)
```

USC Viterbi School of Engineering

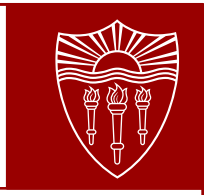
COMPONENT DEFINITION
PERFORM_SMOTE



```python
@component(packages_to_install=["pandas", "numpy", "scikit-learn", "imbalanced-learn==0.11.0"])
def perform_SMOTE(input_df_path:  InputPath('Dataset'),
                  output_df_path: OutputPath('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)

    X = df.drop('TenYearCHD', axis = 1)
    y = df['TenYearCHD']

    # Perform SMOTE oversampling
    smote = SMOTE()
    X_smote, y_smote = smote.fit_resample(X, y)

    # Convert the oversampled feature set and target vector back into a DataFrame
    X_smote_df = pd.DataFrame(X_smote, columns=X.columns)
    y_smote_df = pd.DataFrame(y_smote, columns=['TenYearCHD'])

    # Re-join the features and the target into a single DataFrame
    oversampled_df = pd.concat([X_smote_df, y_smote_df], axis=1)

    # Save the re-joined, oversampled dataset to the specified OutputPath
    oversampled_df.to_csv(output_df_path, index=False)
```

USC Viterbi School of Engineering

COMPONENT DEFINITION
TRAIN_LOGISTIC_REGRESSION



```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib"])
def train_logistic_regression(training_dataset_path: InputPath('Dataset'),
                              trained_model_artifact: Output[Model]):
    import pandas as pd
    from sklearn.linear_model import LogisticRegression
    import joblib
    import os

    train_df = pd.read_csv(training_dataset_path)
    X_train = train_df.drop(['TenYearCHD', 'patientID'], axis=1)
    y_train = train_df['TenYearCHD']

    trained_model = LogisticRegression(max_iter=1000)
    trained_model.fit(X_train, y_train)

    os.makedirs(trained_model_artifact.path, exist_ok=True)
    joblib.dump(trained_model, os.path.join(trained_model_artifact.path, "model.joblib"))
```

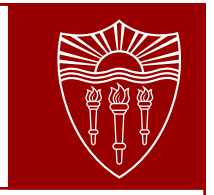
USC Viterbi School of Engineering

COMPONENT DEFINITION
TRAIN_RANDOM_FOREST

```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib"])
def train_random_forest(training_dataset_path: InputPath('Dataset'),
                        trained_model_artifact: OutputPath(Model)):
    import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    import joblib
    import os

    train_df = pd.read_csv(training_dataset_path)
    X_train = train_df.drop(['TenYearCHD', 'patientID'], axis=1)
    y_train = train_df['TenYearCHD']

    trained_model = RandomForestClassifier(n_estimators=500, max_depth=4, random_state=42)
    trained_model.fit(X_train, y_train)

    model_path = os.path.join(trained_model_artifact, "model.joblib")
    os.makedirs(os.path.dirname(model_path), exist_ok=True)
    joblib.dump(trained_model, model_path)
```

USC Viterbi School of Engineering

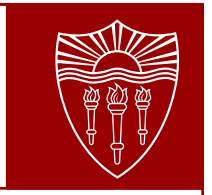
COMPONENT DEFINITION
TRAIN_GRADIENT_BOOSTING



```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib"])
def train_gradient_boosting(training_dataset_path: InputPath('Dataset'),
                            trained_model_artifact: Output[Model]):
    import pandas as pd
    from sklearn.ensemble import GradientBoostingClassifier
    import joblib
    import os


    train_df = pd.read_csv(training_dataset_path)
    X_train = train_df.drop(['TenYearCHD', 'patientID'], axis=1)
    y_train = train_df['TenYearCHD']


    trained_model = GradientBoostingClassifier(n_estimators=500, learning_rate=0.05, max_features = 5,
                                               max_depth=4, random_state=42)
    trained_model.fit(X_train, y_train)


    os.makedirs(trained_model_artifact.path, exist_ok=True)
    joblib.dump(trained_model, os.path.join(trained_model_artifact.path, "model.joblib"))
```

USC Viterbi School of Engineering

COMPONENT DEFINITION
EVALUATE_MODEL

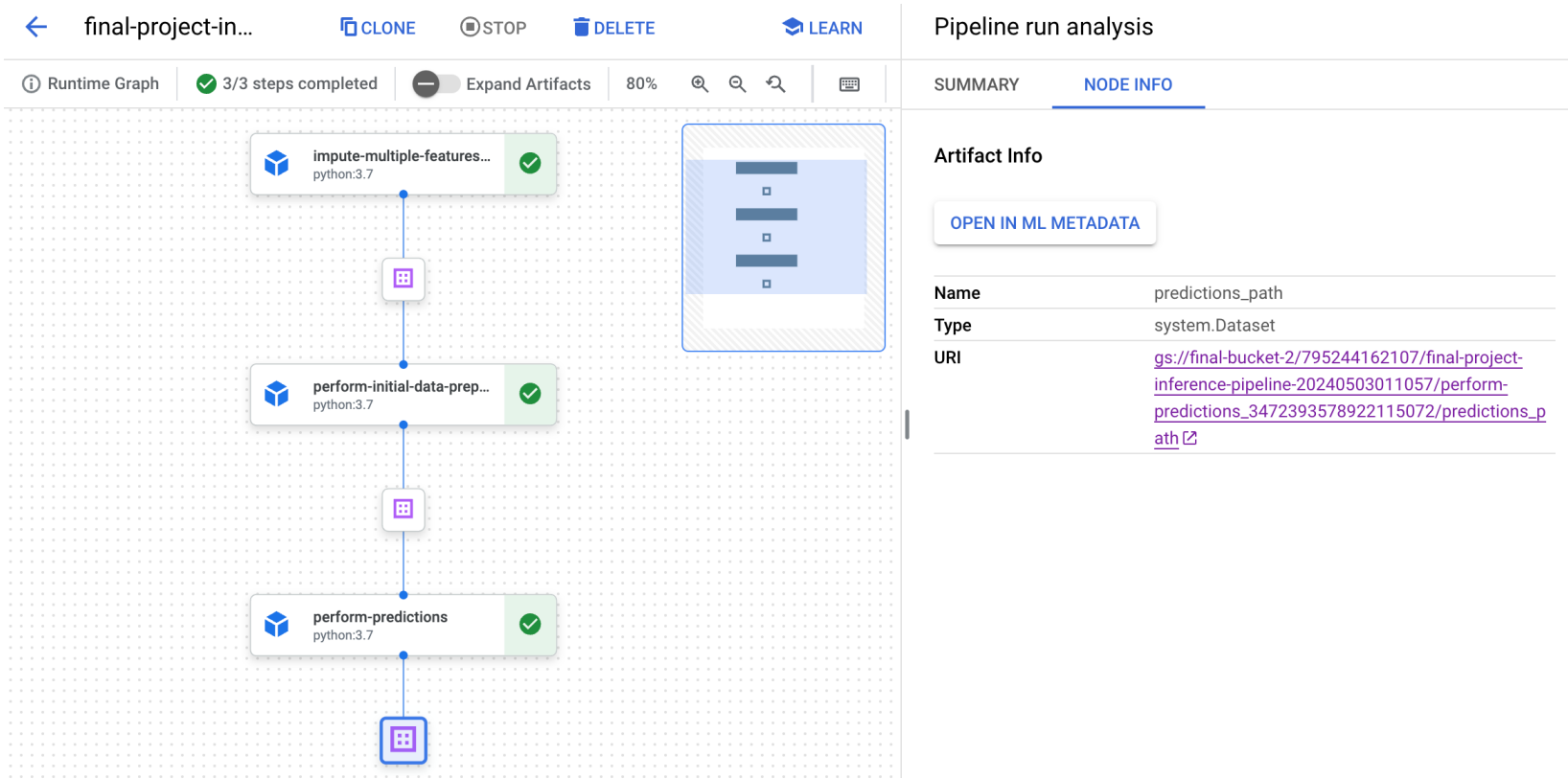


```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib", "numpy"])
def evaluate_model(test_dataset_path: InputPath('Dataset'),
                   model: Input[Model],
                   metrics: Output[Metrics]):
    import pandas as pd
    import joblib
    import numpy as np
    from sklearn.metrics import accuracy_score, f1_score, roc_auc_score


    test_df = pd.read_csv(test_dataset_path)
    X_test = test_df.drop(['TenYearCHD', 'patientID'], axis=1)
    y_test = test_df['TenYearCHD']


    trained_model = joblib.load(model.path + "/model.joblib")

    y_pred = trained_model.predict(X_test)
    y_pred_2 = trained_model.predict_proba(X_test)[:, 1]
    auc_pr = roc_auc_score(y_test, y_pred_2)


    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')


    metrics.log_metric("accuracy", accuracy)
    metrics.log_metric("f1_score", f1)
    metrics.log_metric("AUC_PR", auc_pr)
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

PIPELINE VISUALIZATION



INFERENCE PIPELINE

OTHERS

```
[14] impute_dictionary = pd.read_json(impute_multiple_features_training_artifact_path).to_dict()
     impute_dictionary
```

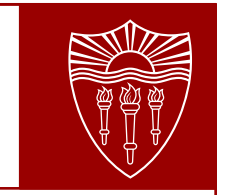

```
{'artifacts': {'feature_medians': {'artifacts': [{'name': 'projects/795244162107/locations/us-central1/metadataStores/default/artifacts/7653179628138035221',
     'uri': 'gs://final-bucket-2/795244162107/final-project-pipeline-20240501182840/impute-multiple-features-training_-3870435342435745792/feature_medians',
     'metadata': {'medians': {'cigsPerDay': 20.0,
       'totChol': 233.0,
       'glucose': 78.0,
       'BMI': 25.38,
       'heartRate': 75.0},
      'modes': {'education': 1.0, 'BPMeds': 0.0}}}]},
  'imputed_dataset_path': {'artifacts': [{'name': 'projects/795244162107/locations/us-central1/metadataStores/default/artifacts/15032303969185358295',
     'uri': 'gs://final-bucket-2/795244162107/final-project-pipeline-20240501182840/impute-multiple-features-training_-3870435342435745792/imputed_dataset_path',
     'metadata': {}}]}}
```


```
training_cigsPerDay_median = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']
training_cigsPerDay_median
```


```
{'medians': {'cigsPerDay': 20.0,
  'totChol': 233.0,
  'glucose': 78.0,
  'BMI': 25.38,
  'heartRate': 75.0},
 'modes': {'education': 1.0, 'BPMeds': 0.0}}
```

INFERENCE PIPELINE

OTHERS



```
medians = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['medians']
modes = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['modes']

median_cigsPerDay = medians['cigsPerDay']
median_totChol = medians['totChol']
median_glucose = medians['glucose']
median_BMI = medians['BMI']
median_heartRate = medians['heartRate']
mode_education = modes['education']
mode_BPMeds = modes['BPMeds']

print("Median of Cigarettes Per Day:", median_cigsPerDay)
print("Median of Total Cholesterol:", median_totChol)
print("Median of Glucose:", median_glucose)
print("Median of BMI:", median_BMI)
print("Median of Heart Rate:", median_heartRate)
print("Mode of Education:", mode_education)
print("Mode of BPMeds:", mode_BPMeds)
```

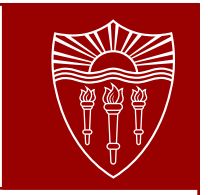
```
Median of Cigarettes Per Day: 20.0
Median of Total Cholesterol: 233.0
Median of Glucose: 78.0
Median of BMI: 25.38
Median of Heart Rate: 75.0
Mode of Education: 1.0
Mode of BPMeds: 0.0
```

```
[ ] # Create a GCS file system object
    fs = gcsfs.GCSFileSystem()

    with fs.open(model_path, 'rb') as f:
        model = joblib.load(f)

    model
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

PIPELINE DEFINITION

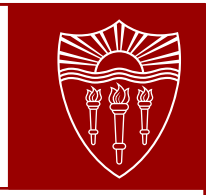


```
from kfp.v2.dsl import pipeline, Output, Dataset
model_path = 'gs://final-bucket-2/795244162107/final-project-pipeline-20240502212050/train-gradient-boosting_-5544868406236282880/trained_model_artifact/model.joblib'
median_cigsPerDay = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['medians']['cigsPerDay']
median_totChol = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['medians']['totChol']
median_glucose = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['medians']['glucose']
median_BMI = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['medians']['BMI']
median_heartRate = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['medians']['heartRate']
mode_education = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['modes']['education']
mode_BPMeds = impute_dictionary['artifacts']['feature_medians']['artifacts'][0]['metadata']['modes']['BPMeds']


@pipeline(name='final_project_inference_pipeline')
def final_project_inference_pipeline(dataset_for_predictions_path: str,
                                     median_cigsPerDay: float = median_cigsPerDay,
                                     median_totChol: float = median_totChol,
                                     median_glucose: float = median_glucose,
                                     median_BMI: float = median_BMI,
                                     median_heartRate: float = median_heartRate,
                                     mode_education: float = mode_education,
                                     mode_BPMed: float = mode_BPMeds,
                                     model_uri: str = model_path):



    imputed_data = impute_multiple_features_validation(
        validation_dataset_path=dataset_for_predictions_path,
        median_cigsPerDay=median_cigsPerDay,
        median_totChol=median_totChol,
        median_glucose=median_glucose,
        median_BMI=median_BMI,
        median_heartRate=median_heartRate,
        mode_education=mode_education,
        mode_BPMeds=mode_BPMeds
    )

    test_data_preparation = perform_initial_data_preparation(input_dataset_path=imputed_data.outputs['imputed_dataset_path'])

    perform_predictions(
        dataset_for_prediction_path=test_data_preparation.outputs['output_dataset_path'],
        model_path=model_uri
    )
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

IMPUTE_MULTIPLE_FEATURES_VALIDATION

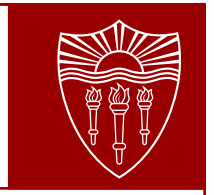


```
@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
def impute_multiple_features_validation(validation_dataset_path: str,
                                        imputed_dataset_path: OutputPath('Dataset'),
                                        median_cigsPerDay: float,
                                        median_totChol: float,
                                        median_glucose: float,
                                        median_BMI: float,
                                        median_heartRate: float,
                                        mode_education: float,
                                        mode_BPMeds: float):
    import pandas as pd
    # Load the test dataset
    df = pd.read_csv(validation_dataset_path)

    import pandas as pd

    df = pd.read_csv(validation_dataset_path)



    df['cigsPerDay'].fillna(median_cigsPerDay, inplace=True)
    df['totChol'].fillna(median_totChol, inplace=True)
    df['glucose'].fillna(median_glucose, inplace=True)
    df['BMI'].fillna(median_BMI, inplace=True)
    df['heartRate'].fillna(median_heartRate, inplace=True)
    df['education'].fillna(mode_education, inplace=True)
    df['BPMeds'].fillna(mode_BPMeds, inplace=True)


    df.to_csv(imputed_dataset_path, index=False)
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

PERFORM_INITIAL_DATA_PREPARATION

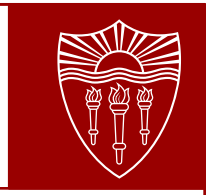


```python
@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
def perform_initial_data_preparation(input_dataset_path: InputPath('Dataset'),
                                     output_dataset_path: OutputPath(Dataset)):
    import pandas as pd
    import numpy as np

    df = pd.read_csv(input_dataset_path)
    df = df.drop(columns=["a1c"])
    df["totChol"] = df["totChol"].clip(upper=700)
    df["BMI"] = df["BMI"].clip(upper=50)
    df["totChol"] = np.log(df["totChol"]+1)
    df["income"] = np.log(df["income"]+1)
    df.loc[df['currentSmoker'] == 0, 'cigsPerDay'] = 0
    df = pd.get_dummies(df, drop_first=True)
    df.to_csv(output_dataset_path, index=False)
```

USC Viterbi School of Engineering

INFERENCE PIPELINE

PERFORM_PREDICTIONS

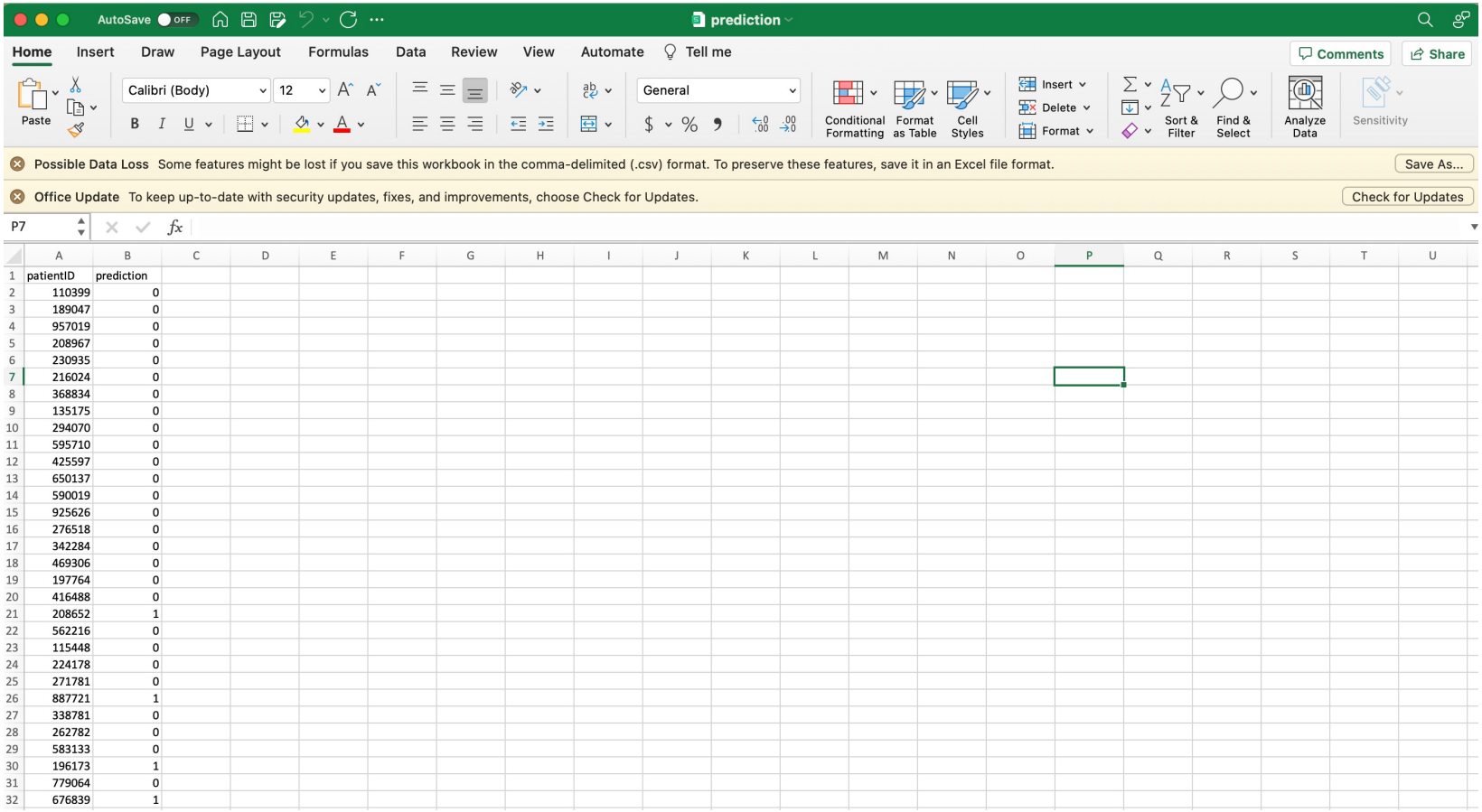


```python
@component(packages_to_install=["pandas", "numpy", "scikit-learn", "joblib", "fsspec", "gcsfs"])
def perform_predictions(dataset_for_prediction_path: InputPath('Dataset'),
                        model_path: str,
                        predictions_path: OutputPath('Dataset')):

    import pandas as pd
    import joblib
    import gcsfs

    # Create a GCS file system object
    fs = gcsfs.GCSFileSystem()

    # Load the trained model
    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_path)

    # Make predictions
    #y_pred = trained_model.predict(pred_df)
    pred_df_2 = pred_df.drop('patientID', axis=1)
    y_pred = trained_model.predict(pred_df_2)
    pred_df['prediction'] = y_pred
    pred_df = pred_df[['patientID', 'prediction']]

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

USC Viterbi School of Engineering

INFERENCE PIPELINE

SCREENSHOT OF PREDICTION CSV

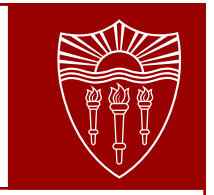


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	patientID	prediction																			
2	110399	0																			
3	189047	0																			
4	957019	0																			
5	208967	0																			
6	230935	0																			
7	216024	0																			
8	368834	0																			
9	135175	0																			
10	294070	0																			
11	595710	0																			
12	425597	0																			
13	650137	0																			
14	590019	0																			
15	925626	0																			
16	276518	0																			
17	342284	0																			
18	469306	0																			
19	197764	0																			
20	416488	0																			
21	208652	1																			
22	562216	0																			
23	115448	0																			
24	224178	0																			
25	271781	0																			
26	887721	1																			
27	338781	0																			
28	262782	0																			
29	583133	0																			
30	196173	1																			
31	779064	0																			
32	676839	1																			

USC Viterbi School of Engineering

INFERENCE PIPELINE

SCREENSHOT OF F1-SCORE



Autograder Results

Results | Code

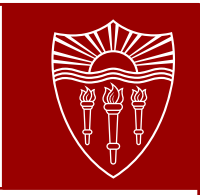
Check submitted files (0/0)

```
Required CSV file submitted.
```

1) F1-Score (79.53/100)

```
Test Failed: 79.53 != 100 : The F1-Score is 0.7953140803491208.
```

SUMMARY DISCUSSION



- SMOTE improved the prediction a lot, because the response variable is unbalanced.
- The logistic regression model appears to not be a good fit for this dataset. Therefore, I try random forest and gradient boosting and it appears that gradient boosting is the best model.
- There are some possible reasons:
  - Gradient boosting typically performs well when dealing with imbalanced data, as it can effectively identify and classify minority classes while reducing overfitting to majority classes.
  - Gradient boosting is an ensemble method.
  - Gradient boosting uses gradient descent to optimize the loss function.