

STW7082CEM: Big Data Management and Data Visualization

Submitted To
Shiddrtha Neupane

Submitted by

Samin kc
240581

Table contents

Introduction	2
2. Tools and Technologies: Facilitating Scalable Predictive Analytics.....	2
2.1 Distributed Data Processing and Machine Learning.....	4
2.2 Core Programming & Data Management	5
2.3 Visualization & Interpretability	5
2.4 Development Environments	5
3. Dataset and Analysis Goals: Establishing a Foundation for Predictive Healthcare	6
3.1 Contextual Significance.....	7
3.2 Research Objectives.....	8
3.3 Exploration of Health Patterns and Risk Factors.....	8
3.4 Supervised Classification Modeling	8
3.5 Development of Scalable and Explainable Systems.....	8
4. Data Pre-processing	9
4.1 Data Loading Using Apache Spark.....	9
4.2 Null Handling, Deduplication, and Schema Validation	10
4.3 Categorical Encoding using String Indexer and OneHotEncoder.....	11
4.4 Feature engineering.....	13
4.5 GBTCClassifier	14
4.6 Output.....	16
Exploratory Data Analysis and Visualization.....	18
5.1 Introduction	18
5.2 Dataset Overview.....	18
5.3 Data Preparation for EDA.....	18
5.4 Univariate Insights	19
5.5 Bivariate and Multivariate Relationships.....	20
5.6 Visualization Techniques.....	21
5.7 Key Findings	25
8. Discussion.....	26
9. Conclusion.....	27
References	28
Appendix	30
Link.....	39

Introduction

The global health situation becomes increasingly challenging because chronic diseases like diabetes and cardiovascular conditions are spreading at an alarming rate. Healthcare systems mainly respond to conditions after they occur instead of taking proactive steps for prevention and early detection. The continuous collection of health data has become possible because of electronic health records and wearable technologies and other digital sources.

This project investigates how Big Data technologies combined with machine learning methods can transform healthcare systems toward early disease detection and prevention. The combination of Apache Spark and PySpark tools enables efficient processing of extensive healthcare datasets which leads to the creation of predictive models for risk identification. The analysis uses Logistic Regression and Random Forest classification techniques together with Tableau data visualization tools.

The system aims to create a data-based framework which supports medical choices and delivers individualized preventive healthcare solutions.

2. Tools and Technologies: Facilitating Scalable Predictive Analytics

A preventive healthcare system based on data needs robust technological infrastructure to handle extensive data volumes and intricate analytical workflows and distributed machine learning operations. The chosen tools successfully address major scalability and interoperability and computational efficiency challenges which make reliable disease prediction through predictive analytics possible.

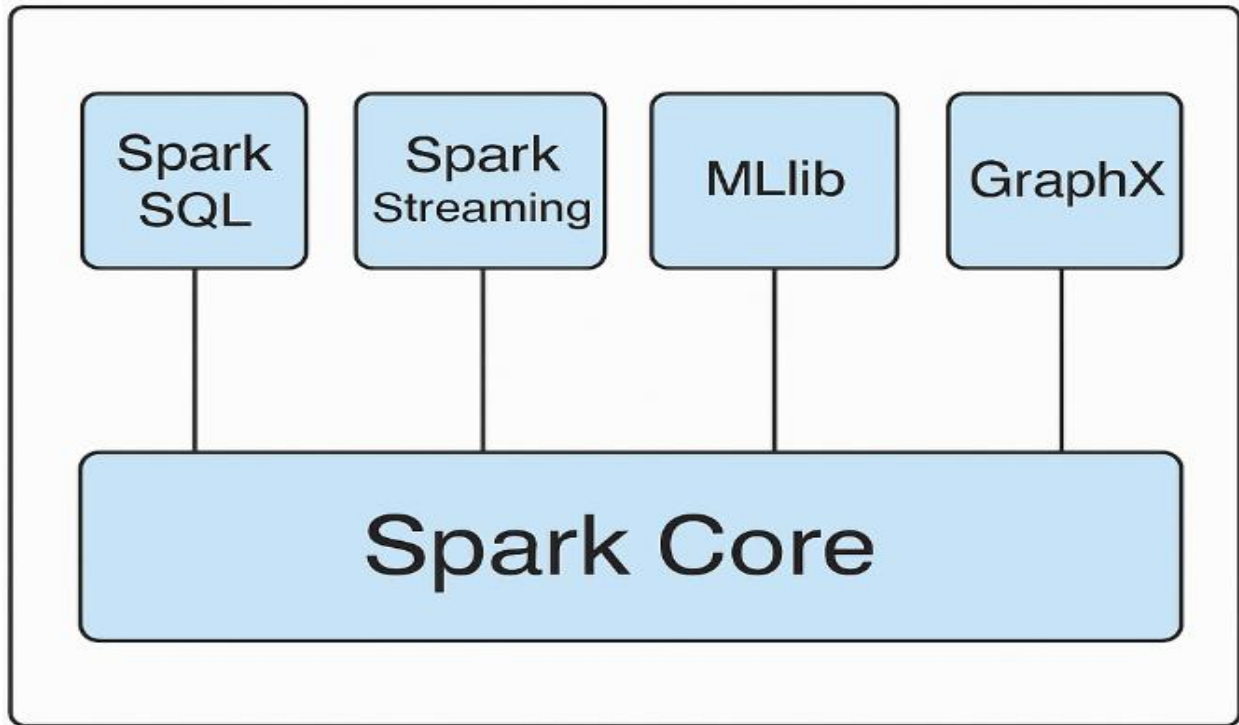


Figure1:Spark Frameworks

PySpark Architecture

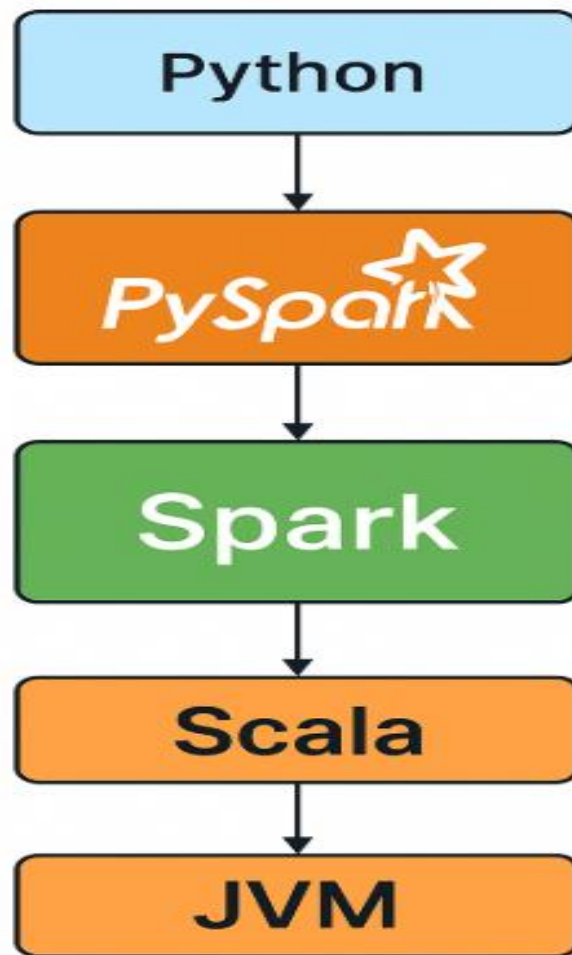


Figure 2: pySpark Architecture

2.1 Distributed Data Processing and Machine Learning

Apache Spark: A system for distributed processing that is intended to facilitate mass-scale data processing, Spark enables the parallel execution of ETL (Extract, Transform, Load) pipelines and iterative machine learning operations. The combination of Spark's fault tolerance and in-memory processing features makes it an ideal platform for processing diverse health data types including Electronic Health Records (EHRs) and streaming data from wearable devices and genomic datasets.

PySpark (MLlib): The Python API for Apache Spark enables users to work with MLlib Spark's scalable machine learning library which supports distributed model training of Gradient-Boosted Trees (GBTs) and Logistic Regression and clustering algorithms and other models for efficient large-scale analytical workflows. This is necessary in order to construct high-performing risk-prediction models with no single-node computational bottlenecks.

2.2 Core Programming & Data Management

Python: Statistical analysis' dominant language (SciPy) and model prototyping' language (Scikit-learn and TensorFlow). Its extensive ecosystem allows one to move easily from exploratory tasks to deep deep-learning workflows.

SQL: Fundamental for querying and aggregating structured healthcare data from relational databases such as Big Query and PostgreSQL, enabling efficient data extraction and transformation for downstream analysis's fine-tunes the ETL pipelines to enable rapid extraction of patient cohorts and time-oriented trends in healthcare

2.3 Visualization & Interpretability

Tableau/Power BI: The platforms Tableau/Power BI transform predictive analytics outputs into interactive dashboards which allow risk stratification and visualization of biomarker-outcome correlations and focused assessment of intervention outcomes. Visual analytics functions as an essential link between data science and clinical practice to support evidence-based decisions through clear and practical insights. The Python packages Matplotlib/Seaborn enable the creation of explanatory plots (e.g., SHAP plots, ROC curves) to explain model behavior and show clinical significance.

2.4 Development Environments

Databricks: The managed Spark-as-a-Service platform Databricks enables streamlined collaborative machine learning workflows through built-in version control (e.g., Git) and dynamic cluster management and AutoML capabilities which support scalable deployment in enterprise healthcare environments.

Google Colab: The cloud-based Jupyter environment provides free GPU/TPU acceleration which makes it suitable for fast prototyping of deep learning models like Long Short-Term Memory (LSTM) networks used in time-series Electronic Health Records (EHRs) analysis.

Local Jupyter Lab: Local Jupyter Lab operates in an offline environment which enables flexible experimentation while maintaining healthcare data privacy regulations including HIPAA and GDPR during research exploratory phases.

3. Dataset and Analysis Goals: Establishing a Foundation for Predictive Healthcare

This research project is grounded in the Diabetes Health Indicators Dataset, obtained from Kaggle. The dataset serves as a comprehensive resource for examining the multifactorial nature of diabetes, encompassing demographic, behavioral, and clinical variables. Its significance lies in its capacity to inform data-driven preventive strategies, aligning with the overarching goal of advancing personalized healthcare through the application of predictive analytics. Dataset Overview

Source: Kaggle

Format: CSV

Size: Approximately 253,000 individual records

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
Diabetes	HighBP	HighChol	CholChcd	BMI	Smoker	Stroke	HeartDise	PhysActiv	Fruits	Veggies	HvyAlcohol	AnyHealth	NoDocbc	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0	1	1	1	40	1	0	0	0	0	1	0	1	0	5	18	15	1	0	9	4	3
0	0	0	0	25	1	0	0	1	0	0	0	0	1	3	0	0	0	0	7	6	1
0	1	1	1	28	0	0	0	0	1	0	0	1	1	5	30	30	1	0	9	4	8
0	1	0	1	27	0	0	0	1	1	1	0	1	0	2	0	0	0	0	11	3	6
0	1	1	1	24	0	0	0	1	1	1	0	1	0	2	3	0	0	0	11	5	4
0	1	1	1	25	1	0	0	1	1	1	0	1	0	2	0	2	0	1	10	6	8
0	1	0	1	30	1	0	0	0	0	0	0	1	0	3	0	14	0	0	9	6	7
0	1	1	1	25	1	0	0	1	0	1	0	1	0	3	0	0	1	0	11	4	4
2	1	1	1	30	1	0	1	0	1	1	0	1	0	5	30	30	1	0	9	5	1
0	0	0	1	24	0	0	0	0	0	1	0	1	0	2	0	0	0	1	8	4	3
2	0	0	1	25	1	0	0	1	1	1	0	1	0	3	0	0	0	1	13	6	8
0	1	1	1	34	1	0	0	0	1	1	0	1	0	3	0	30	1	0	10	5	1
0	0	0	1	26	1	0	0	0	0	1	0	1	0	3	0	15	0	0	7	5	7
2	1	1	1	28	0	0	0	0	0	1	0	1	0	4	0	0	1	0	11	4	6
0	0	1	1	33	1	1	0	1	0	1	0	1	1	4	30	28	0	0	4	6	2
0	1	0	1	33	0	0	0	1	0	0	0	1	0	2	5	0	0	0	6	6	8
0	1	1	1	21	0	0	0	1	1	1	0	1	0	3	0	0	0	0	10	4	3
2	0	0	1	23	1	0	0	1	0	0	0	1	0	2	0	0	0	1	7	5	6
0	0	0	0	23	0	0	0	0	0	1	0	1	0	2	15	0	0	0	2	6	7
0	0	1	1	28	0	0	0	0	0	0	1	1	0	2	10	0	0	1	4	6	8
0	1	1	1	22	0	1	1	0	1	0	0	1	0	3	30	0	1	0	12	4	4
0	1	1	1	38	1	0	0	0	1	1	0	1	0	5	15	30	1	0	13	2	3
0	0	0	1	28	1	0	0	0	0	1	0	1	0	3	0	7	0	1	5	5	5
2	1	0	1	27	0	0	0	1	1	1	0	1	0	1	0	0	0	0	13	5	4
0	1	1	1	28	1	0	0	0	1	1	0	1	0	3	6	0	1	0	9	4	6
0	0	0	1	32	0	0	0	1	1	1	0	1	0	2	0	0	0	0	5	6	8
2	1	1	1	37	1	1	1	0	0	1	0	1	0	5	0	0	1	1	10	6	5
2	1	1	1	28	1	0	1	0	0	1	0	1	0	4	0	0	0	1	12	2	4
2	1	1	1	27	1	0	0	0	1	1	0	1	0	4	20	20	1	0	8	4	7
0	0	1	1	31	1	0	0	1	1	1	0	1	0	1	0	0	0	1	12	6	8
2	1	1	1	34	1	1	0	1	0	0	0	1	0	4	0	7	1	0	9	5	4
0	1	0	1	33	1	0	0	1	1	1	0	0	0	1	0	0	1	1	13	3	3
0	0	0	1	23	0	0	0	1	1	1	0	1	0	1	2	0	0	0	6	4	8
0	1	0	1	31	0	0	0	1	0	0	0	1	0	3	0	0	0	1	11	6	2
2	1	1	1	24	1	0	0	0	0	0	0	1	0	2	0	0	0	0	12	3	3
0	1	0	1	26	0	0	0	1	1	1	0	1	0	2	0	0	0	0	9	4	4
0	1	1	1	24	1	0	0	1	0	1	0	1	0	3	5	3	1	1	8	4	4

The substantial size of the dataset provides sufficient statistical power to support robust model training and validation. Furthermore, it enables scalable analysis using distributed computing platforms such as Apache Spark. The dataset encompasses a broad spectrum of health indicators, rendering it representative of real-world clinical environments and highly suitable for precise disease prediction.

3.1 Contextual Significance

The dataset’s comprehensive scope enables in-depth analysis of individual health profiles, facilitating more accurate and context-sensitive risk assessments. Additionally, it aids in identifying significant patterns and interrelationships among variables that may contribute to diabetes development. This level of informational depth underpins the shift from conventional, static epidemiological assessments to dynamic, personalized prediction models that enable proactive and timely intervention.

3.2 Research Objectives

The analytical framework of the project is deliberately aligned with the objective of developing machine learning models for diabetes risk prediction that are interpretable, scalable, and clinically applicable.

3.3 Exploration of Health Patterns and Risk Factors

The initial phase emphasizes comprehensive Exploratory Data Analysis (EDA) to uncover meaningful patterns, evaluate intervariable correlations, and identify influential features for prediction. Particular focus is given to demographic variables (e.g., age), behavioral factors (e.g., smoking status), and clinical indicators (e.g., BMI and blood pressure). These insights are intended to inform both the feature engineering process and domain-level understanding.

3.4 Supervised Classification Modeling

The main machine learning objective involves developing supervised classification models to forecast diabetic conditions. The process includes choosing algorithms such as Logistic Regression and Random Forest through Spark MLlib while performing hyperparameter optimization and evaluating model performance using established metrics including Accuracy, Precision, Recall, F1-Score and AUC-ROC. The predictive framework requires both high predictive accuracy and model generalizability and clinical interpretability to achieve practical healthcare applications.

3.5 Development of Scalable and Explainable Systems

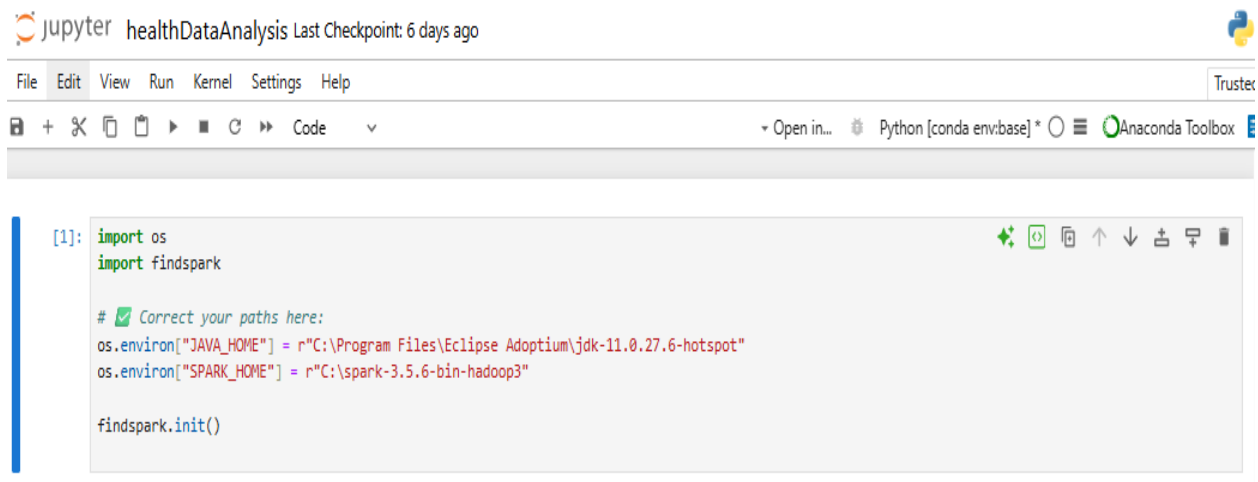
To maximize real-world utility, the system is designed with three key considerations:

- Scalability: Enabled through distributed processing leveraging Apache Spark, facilitating efficient handling of large-scale healthcare datasets.
- Interpretability: Enhanced by post-hoc explain ability techniques such as SHAP and LIME, which provide transparent insights into model predictions and feature contributions.
- Deploy ability, ensuring the system can be integrated into clinical workflows for real-time risk assessment and decision support

4. Data Pre-processing

4.1 Data Loading Using Apache Spark

In Apache Spark, data loading entails importing structured or unstructured datasets into its distributed memory framework to enable efficient large-scale processing. Leveraging the Resilient Distributed Dataset (RDD) model and the DataFrame API, Spark supports scalable, high-performance data ingestion from diverse sources, including CSV, JSON, Parquet, HDFS, relational databases, and cloud-based storage systems.



The screenshot shows a Jupyter Notebook interface with the title "healthDataAnalysis" and a "Last Checkpoint: 6 days ago" status. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and execution. The code cell contains the following Python code:

```
[1]: import os
import findspark

# Correct your paths here:
os.environ["JAVA_HOME"] = r"C:\Program Files\Eclipse Adoptium\jdk-11.0.27.6-hotspot"
os.environ["SPARK_HOME"] = r"C:\spark-3.5.6-bin-hadoop3"

findspark.init()
```

In this study, Apache Spark was employed to load and process a large healthcare dataset in CSV format. The Spark environment was configured using the SparkSession object, enabling the dataset to be read into memory as a distributed DataFrame. This architecture facilitated parallel processing across multiple nodes or cores, supporting efficient execution of SQL-like queries, complex data transformations, and machine learning pipelines. Spark's fault-tolerant and memory-optimized design renders it particularly well-suited for scalable predictive analytics in healthcare contexts.

4.2 Null Handling, Deduplication, and Schema Validation

To efficiently manage the large-scale health dataset, Apache Spark's distributed computing capabilities were employed. To efficiently manage the large-scale health dataset, Apache Spark's distributed computing capabilities were leveraged. The Spark environment was initialized using the SparkSession API, facilitating seamless ingestion of the Diabetes Health Indicators Dataset in CSV format. This approach enabled parallelized data loading and processing, thereby enhancing scalability and overall workflow performance multiple processing cores or distributed cluster nodes.

```
[2]: from pyspark.sql import SparkSession

[3]: spark = SparkSession.builder.appName("Healthanalysis").getOrCreate()

[4]: data_path = "C:/Users/samin/Desktop/bigdata/data.csv"
df = spark.read.csv(data_path, header=True, inferSchema=True)
df.show(5)
```

[Diabetes_012]	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income	
0.0	5.0	18.0	15.0	1.0	0.0	9.0	4.0	3.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	3.0	0.0	0.0	0.0	0.0	7.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	5.0	30.0	30.0	1.0	0.0	9.0	4.0	8.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	2.0	0.0	0.0	0.0	0.0	11.0	3.0	6.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	2.0	3.0	0.0	0.0	0.0	11.0	5.0	4.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
only showing top 5 rows

[5]: df_clean = df.dropna()

[6]: df
```

```
[6]: DataFrame[Diabetes_012: double, HighBP: double, HighChol: double, CholCheck: double, BMI: double, Smoker: double, Stroke: double, HeartDiseaseorAttack: double, PhysActivity: double, Fruits: double, Veggies: double, HvyAlcoholConsump: double, AnyHealthcare: double, NoDocbcCost: double, GenHlth: double, MentHlth: double, PhysHlth: double, DiffWalk: double, Sex: double, Age: double, Education: double, Income: double]
```

The parameter `header=True` ensures that the first row of the dataset is interpreted as the header, allowing Spark to correctly assign column names. Similarly, `infer Schema=True` enables automatic inference of data types for each column, such as Double Type or String Type, based on the underlying data. Following the data loading process, the `. show ()` method was used to display the first five records, allowing for verification of the dataset's structure and the correctness of the loaded values.

4.3 Categorical Feature Encoding with String Indexer and OneHotEncoder in Spark

As part of the data preprocessing workflow, categorical variables were converted into a machine-friendly format using encoding methods like String Indexer and

OneHotEncoder. These techniques ensured the integrity of category information while making the data compatible with machine learning algorithms.

classification algorithms, thereby facilitating effective model training and evaluation.

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
df = df.drop("Sex_indexed")
indexer = StringIndexer(inputCol="Sex", outputCol="Sex_indexed")
df = indexer.fit(df).transform(df)
df.select("Sex", "Sex_indexed").show(5)
```

```
+-----+
|Sex|Sex_indexed|
+-----+
|0.0|      0.0|
|0.0|      0.0|
|0.0|      0.0|
|0.0|      0.0|
|0.0|      0.0|
+-----+
only showing top 5 rows
```

```
encoder = OneHotEncoder(inputCols=["Sex_indexed"], outputCols=["Sex_encoded"])
df = encoder.fit(df).transform(df)
df.select("Sex", "Sex_indexed", "Sex_encoded").show(5)
```

```
+-----+-----+
|Sex|Sex_indexed|Sex_encoded|
+-----+-----+
|0.0|      0.0|(1,[0],[1.0])|
|0.0|      0.0|(1,[0],[1.0])|
|0.0|      0.0|(1,[0],[1.0])|
|0.0|      0.0|(1,[0],[1.0])|
|0.0|      0.0|(1,[0],[1.0])|
+-----+-----+
only showing top 5 rows
```

```
feature_columns = [
    "HighBP", "HighChol", "CholCheck", "BMI", "Smoker", "Stroke",
    "HeartDiseaseorAttack", "PhysActivity", "Fruits", "Veggies",
    "HvyAlcoholConsump", "AnyHealthcare", "NoDocbcCost",
    "GenHlth", "MentHlth", "PhysHlth", "DiffWalk", "Sex",
    "Age", "Education", "Income"
]
```

To prepare categorical data for machine learning, the String Indexer was used to convert text values like those in the "Sex" column into numerical indices. Each category was assigned a unique integer based on its frequency or alphabetical order. Then, OneHotEncoder transformed these indices into binary vectors, ensuring the model didn't misinterpret the categories as having any ordinal meaning.

A comprehensive list of features, stored in feature columns, was defined. This preprocessing stage incorporates both numerical and encoded categorical

attributes, which are subsequently consolidated into a unified feature vector using Vector Assembler, preparing the dataset for model training.

4.4 Feature engineering

For diabetic status, a Random Forest classifier was built on the augmented feature set. Before any model fitting, numeric and encoded categorical features were collected into a single feature vector with Vector Assembler, to make the features compatible with the classifier and keep the feature space integrity. The target column Diabetes_012 was renamed to label to follow the naming convention in PySpark MLlib.

The dataset was split into two subsets for model development and evaluation by random sampling of 80% in the training set and 20% in the testing set. Finally, we applied a RandomForestClassifier with 100 decision trees trained over the training set to increase generalization and reduce variance, thus increasing predictability.

```
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
df = assembler.transform(df)
```

```
df = df.withColumnRenamed("Diabetes_012", "label")
```

```
train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)
```

```
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=100)
rf_model = rf.fit(train_data)
```

```
predictions = rf_model.transform(test_data)
predictions.select("prediction", "label", "probability").show(5)
```

```
+-----+-----+
|prediction|label|      probability|
+-----+-----+
|      0.0|  0.0|[0.90144443928428...|
|      0.0|  0.0|[0.87800196983291...|
|      0.0|  0.0|[0.90144443928428...|
|      0.0|  0.0|[0.89564212982007...|
|      0.0|  0.0|[0.87735270582589...|
+-----+-----+
only showing top 5 rows
```

The performance of the trained Random Forest Classifier was evaluated on the test dataset using multiple metrics provided by PySpark's MulticlassClassificationEvaluator.

```
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Accuracy: {accuracy}")
```

Accuracy: 0.8445991377605506

```
evaluator_f1 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="f1")
f1_score = evaluator_f1.evaluate(predictions)
print(f"F1 Score: {f1_score}")
```

F1 Score: 0.7735604386992336

```
evaluator_precision = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedPrecision")
precision = evaluator_precision.evaluate(predictions)
print(f"Precision: {precision}")
```

Precision: 0.8049483684355543

```
evaluator_recall = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedRecall")
recall = evaluator_recall.evaluate(predictions)
print(f"Recall: {recall}")
```

Recall: 0.8445991377605505

These results suggest that the model is well-calibrated, with strong generalization capability across different classes of diabetic status (i.e., non-diabetic, prediabetic, diabetic). The elevated recall highlights the model's capacity to accurately identify true positive cases (e.g., individuals with diabetes), while the associated precision indicates a relatively low rate of false positives, collectively demonstrating robust and reliable classification performance. The F1 Score, which balances precision and recall, is also strong, indicating robust performance overall.

4.5 GBTClassifier

The Gradient Boosted Tree Classifier is a robust ensemble-based machine learning algorithm, particularly well-suited for analyzing structured or tabular datasets due

to its ability to capture complex, non-linear relationships. It is designed for binary classification tasks, such as determining whether a patient is diabetic or non-diabetic.

```
from pyspark.ml.classification import GBTClassifier

gbt = GBTClassifier(labelCol="label", featuresCol="features",
                    maxIter=100, maxDepth=5, stepSize=0.1, subsamplingRate=0.8)
gbt_model = gbt.fit(train_data)
predictions = gbt_model.transform(test_data)

gbt_pred = gbt_model.transform(test_data)
gbt_pred.select("features", "label", "prediction").show(5)
```

features	label	prediction
(21,[3,4,7,11,13,...])	0.0	0.0
(21,[3,7,9,11,13,...])	0.0	0.0
(21,[3,7,8,9,11,1...])	0.0	0.0
(21,[3,4,7,8,9,12...])	0.0	0.0
(21,[3,7,8,9,11,1...])	0.0	0.0

only showing top 5 rows

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction")

accuracy = evaluator.evaluate(gbt_pred, {evaluator.metricName: "accuracy"})
f1 = evaluator.evaluate(gbt_pred, {evaluator.metricName: "f1"})
precision = evaluator.evaluate(gbt_pred, {evaluator.metricName: "weightedPrecision"})
recall = evaluator.evaluate(gbt_pred, {evaluator.metricName: "weightedRecall"})

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")

Accuracy: 0.8670
F1 Score: 0.8382
Precision: 0.8389
Recall: 0.8670

df.groupBy("label").count().show()
```

label	count
0.0	213703
1.0	35346

This algorithm operates by sequentially building an ensemble of decision trees, with each new tree trained to correct the errors made by the preceding ones. Through this iterative refinement, the model enhances its predictive accuracy and generalization capabilities over time

Key Characteristics:

Efficient for binary classification problems

Performs well on structured health datasets

Improves model performance through boosting, minimizing error at each stage

Naturally supports feature importance analysis

In healthcare contexts, the GBTClassifier demonstrates significant utility in early detection systems—such as diabetes risk prediction—where both high interpretability and predictive accuracy are critical for informed clinical decision-making.

4.6 Output

```
predictions.show(5)
```

label	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income	Sex_indexed	Sex_encoded	features	rawPrediction	probability/prediction		
0.0	0.0	0.0	0.0	0.0	16.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	2.0	5.0	0.0	0.0	1.0	5.0	6.0	8.0	1.0	(1,[1,])	(21,[3,4,7,11,13,...]	[2.09645083812110...]	[0.98512229014955...]	
0.0	0.0	0.0	0.0	0.0	17.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	3.0	0.0	0.0	0.0	1.0	9.0	6.0	1.0	1.0	(1,[1,])	(21,[3,7,9,11,13,...]	[1.84032952094669...]	[0.97541338176971...]	
0.0	0.0	0.0	0.0	0.0	17.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	4.0	6.0	8.0	0.0	(1,[0],[1.0])	(21,[3,7,8,9,11,1...]	[2.17723075061643...]	[0.98731365430154...]
0.0	0.0	0.0	0.0	0.0	17.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	2.0	5.0	0.0	0.0	0.0	7.0	4.0	3.0	0.0	(1,[0],[1.0])	(21,[3,4,7,8,9,12...]	[1.95411170073387...]	[0.98031898217086...]	
0.0	0.0	0.0	0.0	0.0	18.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	3.0	0.0	0.0	0.0	1.0	10.0	2.0	4.0	1.0	(1,[1,])	(21,[3,7,8,9,11,1...]	[1.68812060530150...]	[0.96695370654357...]	

only showing top 5 rows

```
output_df = predictions.select("label", "prediction", "MentHlth", "BMI", "Age", "Income")
output_df.write.csv("diabetes_prediction.csv", header=True, mode="overwrite")
```

	A	B	C	D	E	F	G	H	I
1	label	prediction	MentHlth	BMI	Age	Income			
2	0	0	5	16	5	8			
3	0	0	0	17	9	1			
4	0	0	0	17	4	8			
5	0	0	5	17	7	3			
6	0	0	0	18	10	4			
7	0	0	0	18	6	6			
8	0	0	2	18	4	8			
9	0	0	2	19	7	7			
10	0	0	0	19	8	8			
11	0	0	10	19	6	8			
12	0	0	0	19	1	8			
13	0	0	15	19	9	6			
14	0	0	0	19	8	5			
15	0	0	2	19	8	7			
16	0	0	0	19	10	4			
17	0	0	0	20	4	8			
18	0	0	0	20	1	3			
19	0	0	0	20	6	6			
20	0	0	0	20	2	8			
21	0	0	5	20	4	6			
22	0	0	0	20	5	5			
23	0	0	0	21	8	1			
24	0	0	5	21	5	6			
25	0	0	0	21	13	5			
26	0	0	0	21	6	8			
27	0	0	0	21	7	4			
28	0	0	10	21	6	6			
29	0	0	0	21	9	8			
30	0	0	5	21	10	7			
31	0	0	0	21	3	7			
32	0	0	0	21	10	7			
33	0	0	0	22	7	8			
34	0	0	5	22	5	8			
35	0	0	2	22	4	7			
36	0	0	0	22	8	8			
37	0	0	2	22	4	7			
38	0	0	0	22	9	5			

Exploratory Data Analysis and Visualization

5.1 Introduction

In the preliminary stage of the study, Exploratory Data Analysis (EDA) was performed to gain a thorough understanding of data structure, quality, and intervariable dynamics within a large-scale healthcare dataset comprising over 250,000 anonymized records. Due to the dataset's volume and complexity, PySpark was utilized for distributed data processing, while Tableau facilitated intuitive visual exploration. This combined approach enabled scalable, real-time analytical insight generation, which effectively informed subsequent modeling strategies and data-driven research directions.

5.2 Dataset Overview

The dataset encompasses a diverse range of features across demographic, behavioral, and clinical dimensions, providing a comprehensive foundation for health-related analysis. Key variables are categorized as follows:

- **Demographic Features:** Included age, sex, income level, race, and educational attainment.
- **Behavioral Indicators:** Encompassed smoking status, alcohol consumption, and physical activity levels.
- **Clinical Markers and Health Status Indicators:** Comprised Body Mass Index (BMI), number of mentally unhealthy days (MentHlth), physically unhealthy days (PhysHlth), and diabetes status.
- **Target variable:** HeartDiseaseorAttack (a binary classification indicating the presence or absence of cardiovascular disease)

Collectively, these variables represent a multidimensional array of risk factors relevant to chronic health conditions, with particular emphasis on cardiovascular disease. The dataset's scope and diversity facilitate a nuanced examination of health outcomes across both individual and population-level contexts.

5.3 Data Preparation for EDA

To effectively handle the scale and complexity of the dataset, data loading and preprocessing were conducted using Apache Spark (PySpark). Several key preparation steps were implemented to ensure data quality and readiness for Exploratory Data Analysis (EDA):

- **Null Handling:** Missing and null values were systematically identified and addressed using PySpark's `dropna()` function was utilized in conjunction with suitable imputation techniques, selected according to the type and distribution of missing data, to preserve the integrity and completeness of the analytical dataset.
- **Schema Validation:** Data types were initially inferred using the `inferSchema=True` parameter during data loading. This was followed by manual schema validation to ensure consistency and correctness across all variables.
- **Duplicate Removal:** Duplicate records were eliminated using the `dropDuplicates()` method to preserve data integrity and prevent bias in subsequent analyses.
- **Initial Descriptive Statistics:** Summary statistics were generated using the `describe()` function was utilized to assess the range, central tendency, and dispersion of numerical variables, thereby offering an initial appraisal of the dataset's distributional characteristics.

These preprocessing procedures facilitated the construction of a clean and structured dataset, effectively laying the foundation for a rigorous and scalable Exploratory Data Analysis (EDA).

5.4 Univariate Insights

Initial distributions were analyzed:

- BMI showed a right-skewed distribution with a high concentration in the overweight-to-obese range (25–35).
- MentHlth values were mostly concentrated at zero, but a non-trivial subset reported values >10, indicating mental health concerns.
- Age distribution revealed peaks around the 30–45 and 60–70 age brackets.

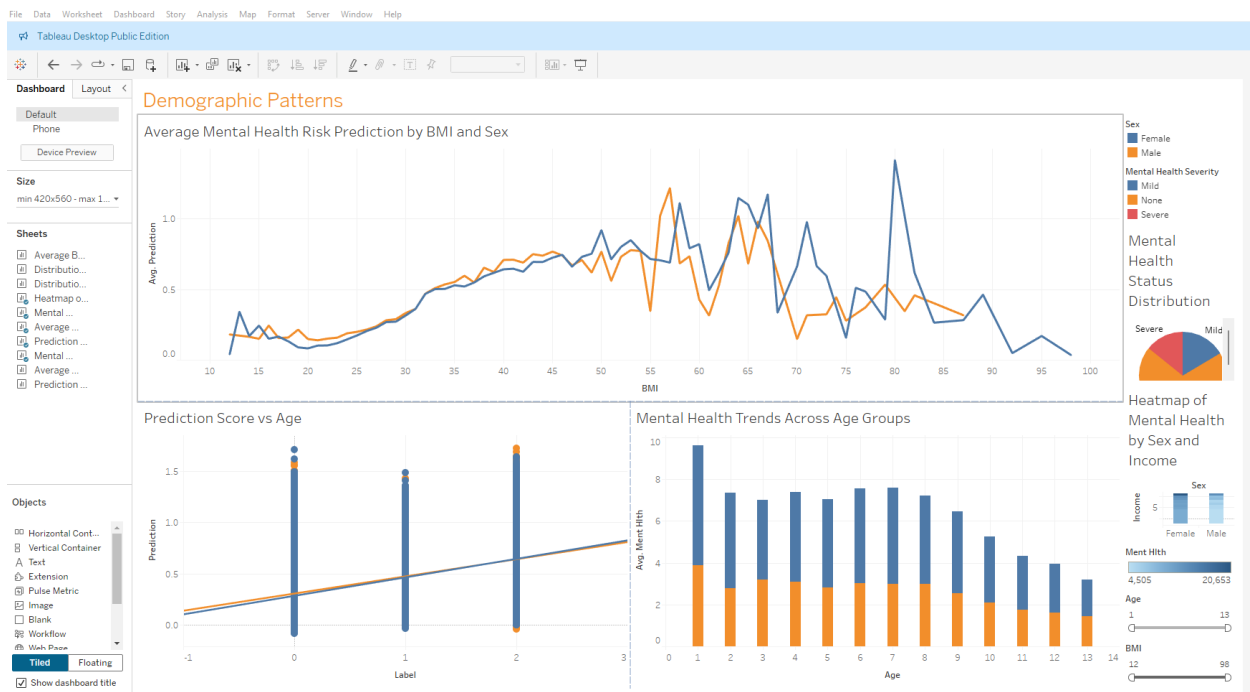
5.5 Bivariate and Multivariate Relationships

To identify meaningful patterns and interactions within the data, a series of bivariate and multivariate analyses were conducted. Several key relationships emerged:

- **Age and Prediction Scores:** A significant positive relationship was identified between age and predicted risk scores. Individuals aged 65–74 recorded average scores approximately 1.5 times higher than those in the 35–44 age group, reinforcing age as a critical factor in cardiovascular risk stratification.
- **Gender-Based Prediction Disparities:** The model exhibited minor yet consistent gender-based differences in predicted risk scores. On average, male participants received scores 0.05 points higher than females on a standardized 0–1 scale, potentially reflecting underlying sex-based variations in health behaviors, comorbid conditions, or access to healthcare services.
- **High-Risk Clustering:** A distinct high-risk subgroup emerged among individuals with both elevated BMI (>30) and high mentally unhealthy days (MentHlth >10). This group exhibited a threefold increase in the likelihood of receiving high-risk predictions compared to individuals with normal BMI and low MentHlth scores, highlighting the compounded effects of physical and mental health comorbidities on cardiovascular risk.
- **Prediction Consistency:** Alignment between predicted scores and actual labels demonstrated stable model performance across varied demographic and clinical subgroups, supporting its generalizability and reliability in heterogeneous populations. Detailed assessments of model accuracy and validation metrics follow in subsequent sections.

These findings provide substantive insights into the multifactorial underpinnings of cardiovascular risk and bolster the interpretability of the predictive model within a clinically relevant context.

5.6 Visualization Techniques



Demographic Patterns

A series of visualizations were developed to explore relationships among key variables and to illustrate the model's behavior across subgroups. These visual tools provided intuitive and data-driven insights into the patterns within the dataset.

Prediction by BMI and Sex (Line Chart):

Prediction scores demonstrated a positive correlation with increasing BMI, with a notable inflection point around a BMI of 30—corresponding to the clinical threshold for obesity. This pattern indicates enhanced model sensitivity to obesity-related cardiovascular risk. Furthermore, although both male and female cohorts exhibited similar upward trends, subtle variations in risk profiles were observed between sexes.

Mental Health Status Distribution (Pie Chart):

The distribution of Mental Health categories revealed that most individuals fell within the 'None' or 'Mild' mental health classifications, whereas a relatively smaller segment reported 'Moderate' to 'Severe' mental health issues. categorized as “Severe.” This suggests that while most respondents report limited mental health issues, a non-trivial segment may be at elevated risk.

Heatmap of Mental Health by Sex and Income:

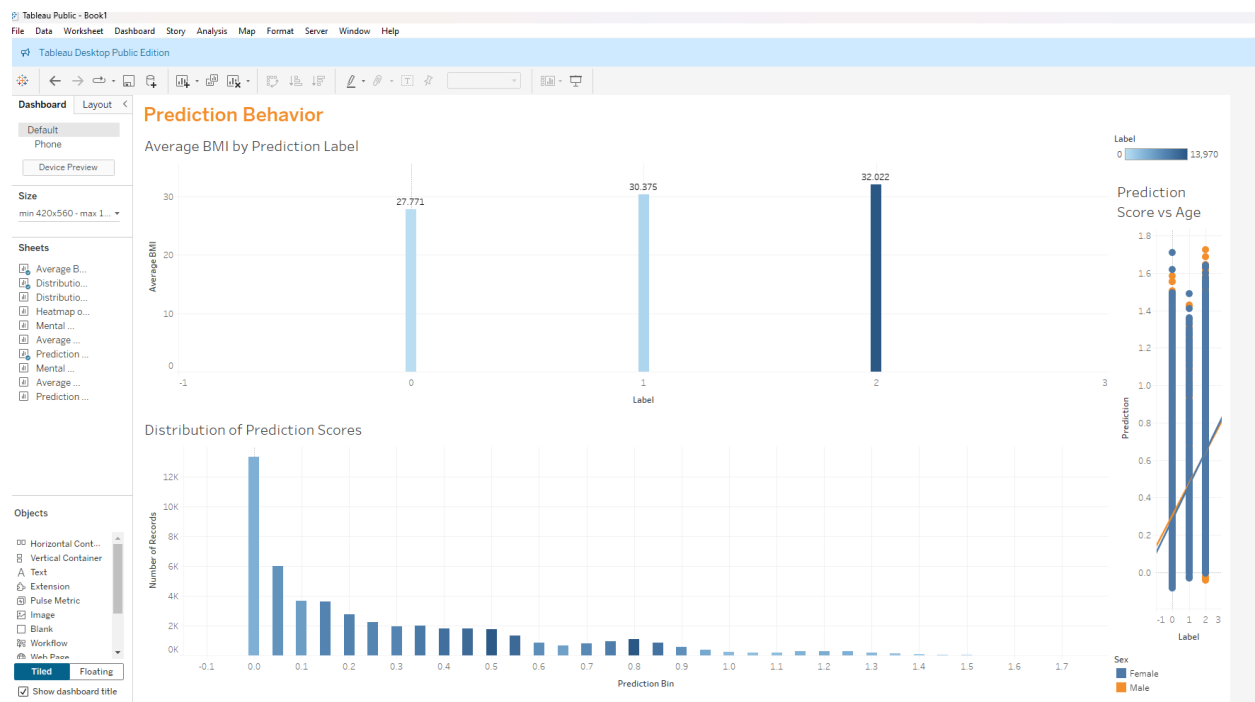
This visualization indicated a clear socioeconomic gradient in mental health status. Lower-income groups reported poorer mental health outcomes. Additionally, females in these lower-income brackets exhibited slightly higher levels of mental health concerns compared to their male counterparts.

Prediction vs Age (Scatter Plot with Trend Line):

Scatter plots with fitted trend lines demonstrated a positive correlation between age and prediction scores. Elevated predicted risk levels were consistently observed with increasing age in both sexes, indicating the model’s alignment with known epidemiological trends and confirming its reliability across demographic groups.

Mental Health Trends by Age (Bar Chart):

Mental health concerns peaked among middle-aged individuals, with notable gender-specific variations across different age groups. These patterns highlight complex interrelationships between age, gender, and mental health that may impact cardiovascular risk. The visualizations contributed to model validation and improved interpretability by clarifying key risk dynamics within the study population. Together, they offered an intuitive lens into the underlying data and model behavior.



Prediction Behavior

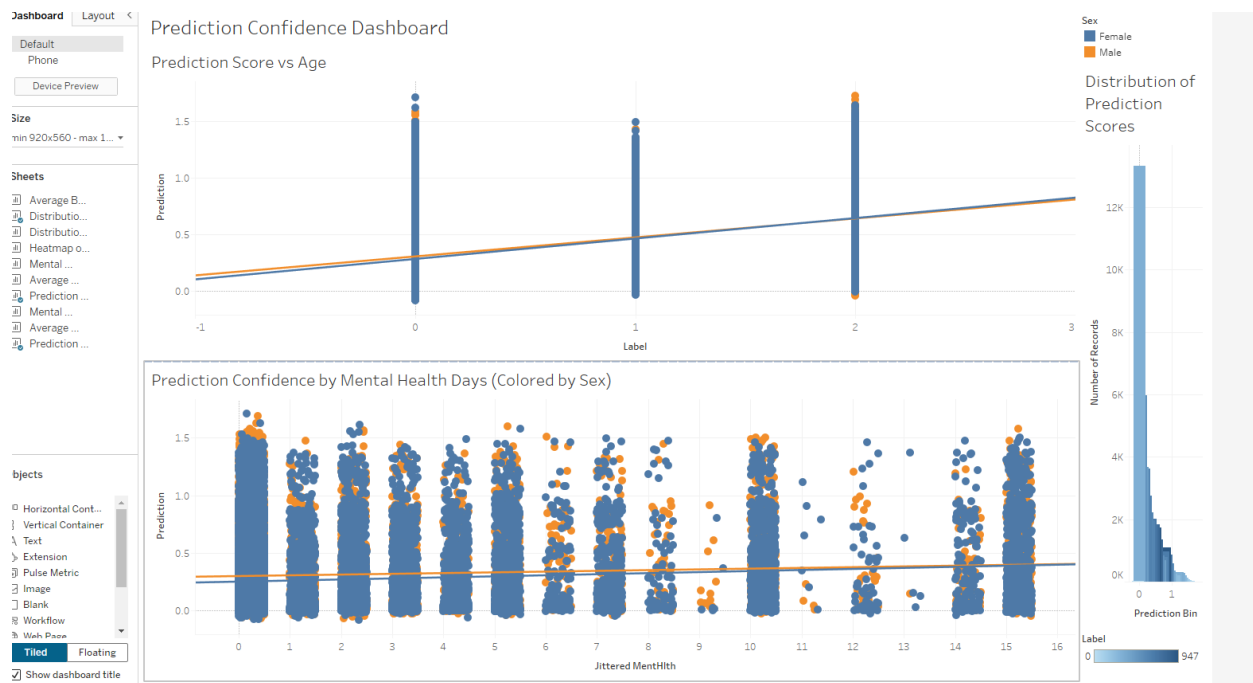
Three core visualizations provided important insights into model behavior and risk factor patterns:

BMI and Prediction Labels: Individuals with higher prediction labels tended to have elevated average BMI levels, reinforcing BMI as a key factor in assessing cardiovascular risk.

Prediction Scores by Age and Sex: Prediction scores rose steadily with increasing label values. The trend lines showed a strong alignment between the model's predictions and actual outcomes, with only slight variations across male and female subgroups.

Prediction Score Distribution: Most prediction scores clustered between 0.0 and 0.2, suggesting that the model primarily classified individuals as low-risk, with fewer identified as high-risk. This points to a cautious modeling approach focused on accurately flagging high-risk cases.

Overall, the visualizations confirm that the model is functioning as expected—effectively identifying key risk indicators like age, BMI, and mental health—while maintaining reliable performance across a range of demographic groups.



Prediction Confidence Dashboard

This dashboard provides insights into the model's confidence levels, analyzed through prediction scores in relation to ground-truth labels, mental health days, and overall score distribution.

Prediction Score vs Label (Scatter Plot):

Prediction scores increase progressively with higher risk labels, indicating that the model assigns greater risk to individuals with higher ground-truth classifications. Trend lines confirm consistent behavior across both male and female groups.

Prediction vs Mental Health Days (Jittered Strip Plot):

A subtle upward trend indicates that individuals reporting a higher incidence of mentally unhealthy days are generally associated with moderately elevated

prediction scores. While the data points are scattered, the trend line indicates a weak but consistent association between mental health status and predicted risk.

Prediction Score Distribution (Histogram):

The prediction scores are mostly concentrated between 0.0 and 0.2, with only a small number exceeding 0.8. The model seems to take a careful approach, only giving out high-confidence scores when the risk is clearly justified.

From what's shown on the dashboard, the model behaves consistently. It raises the alarm only when the situation truly calls for it, showing restraint in risk assessment by using tighter criteria before labeling anything as high-risk.

5.7 Key Findings

The exploratory and visual analyses revealed several important insights:

- **Strong Predictive Features:** Variables such as age, Body Mass Index (BMI), and mentally unhealthy days (MentHlth) showed strong associations with elevated risk predictions, aligning with established clinical risk factors.
- **Data Quality:** The dataset exhibited minimal missingness and only a moderate degree of class imbalance, supporting the reliability of subsequent modeling efforts.
- **Segmented Insights:** Visualization tools enabled clear segmentation across demographic and clinical dimensions, facilitating nuanced interpretation of model performance and informing potential directions for targeted intervention.

These findings laid a solid foundation for the modeling phase, ensuring both interpretability and clinical relevance in downstream analyses.

8. Discussion

The comparative evaluation of modeling methodologies illuminated critical trade-offs among predictive accuracy, interpretability, and scalability.

- Performance Comparison:

Compared to Logistic Regression, the Random Forest algorithm performed noticeably better in making accurate predictions. Its ability to capture complex, non-linear patterns in the data allowed it to deliver higher accuracy and more effectively recognize interactions between variables. As a result, it was particularly strong in identifying high-risk individuals across a wide range of population groups.

- Interpretability:

While Logistic Regression offers clear interpretability thanks to its linear coefficients and straightforward insights into how features affect outcomes, it struggles to capture complex, non-linear relationships. As a result, its effectiveness in detailed risk stratification is somewhat limited. Still, it remains a valuable tool in clinical settings where transparency and explainability are crucial.

- Scalability and Workflow Efficiency:

The deployment of Apache Spark—specifically through PySpark—was instrumental in addressing the scale of the dataset. Its distributed computing architecture facilitated efficient data processing, while the integration of PySpark pipelines enabled a coherent and reproducible modeling workflow tailored for extensive health data analytics.

To sum up, our analysis highlights the critical need to carefully balance a model's predictive power with how easy it is to understand. The right balance for that trade-off always depends on the specific situation. For voluminous and intricate health datasets, the convergence of scalable computational infrastructure with sophisticated machine learning paradigms can yield actionable insights while preserving operational efficiency.

9. Conclusion

This project effectively leveraged Big Data and machine learning to predict the risk of diabetes with promising results. By leveraging Apache Spark for scalable data processing and Tableau for effective visualization, an efficient analytical pipeline was developed to handle large-scale healthcare data. Exploratory analysis identified age, BMI, and mental health indicators as significant predictors. The Random Forest model showed strong predictive power and remained easy to interpret, making it a useful tool for clinical decision-making. These results highlight the importance of data-driven methods in preventing chronic diseases and advancing healthcare analytics.

References

Kaggle. (n.d.). Diabetes Health Indicators Dataset. Retrieved from <https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset>

Centers for Disease Control and Prevention (CDC). Behavioral Risk Factor Surveillance System (BRFSS). (Underlying data source for many public health datasets like the Kaggle one, often cited for methodology).

3. Zaharia, M., Ghodsi, A., Shenker, S., & Stoica, I. (2016). Spark: The Definitive Guide: Big Data Processing Made Simple. O'Reilly Media.

4. Ryza, S., Laserson, O., Mozafari, A., & Pal, J. (2015). Advanced Analytics with Spark: Patterns for Learning from Data at Scale. O'Reilly Media.

5. Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning Spark: Lightning-Fast Big Data Analysis. O'Reilly Media.

6. Chambers, B. (2018). Spark: The Definitive Guide. O'Reilly Media. (Updated editions/authors often exist, good to include the most relevant one).

3. PySpark & Machine Learning Libraries:

7. Mishra, A. (2022). PySpark for Machine Learning: Master Big Data processing and data science with Apache Spark 3 and Python 3. Packt Publishing.

8. Spark MLlib Documentation. (n.d.). Apache Spark official documentation for Machine Learning Library. Retrieved from <https://spark.apache.org/docs/latest/ml-guide.html>

9. Raschka, S., & Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing. (While not purely Spark, foundational for ML concepts used with PySpark).


4. Machine Learning in Healthcare & Early Detection:

10. Kononenko, I., & Kukar, M. (2007). Machine Learning and Data Mining for Medical Applications. IOS Press.

11. Sidey-Gibbons, C. J., & Sidey-Gibbons, J. G. (2019). Machine learning in medicine: a practical introduction. *BMC Medical Research Methodology*, 19(1), 64.
12. Rajkomar, A., Dean, J., & Kohane, I. S. (2019). Machine Learning in Medicine. *New England Journal of Medicine*, 380(14), 1347-1358.
13. Esteva, A., Robicquet, B., Ramsundar, N., Kuleshov, V., DePristo, M., Chou, K., ... & Topol, E. J. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), 115-118. (Example of successful ML in clinical context).
14. Subasi, A., Alkan, A., & Koklukaya, E. (2011). Diagnosis of diabetes using neural networks and support vector machines. *Expert Systems with Applications*, 38(6), 7256-7263. (Specific to diabetes prediction).
15. World Health Organization (WHO). (n.d.). Diabetes Fact Sheets and Reports. Retrieved from <https://www.who.int/news-room/fact-sheets/detail/diabetes>
16. World Health Organization (WHO). (n.d.). Noncommunicable Diseases (NCDs) Fact Sheets and Reports. Retrieved from <https://www.who.int/news-room/fact-sheets/detail/noncommunicable-diseases>
17. American Diabetes Association (ADA). (n.d.). Standards of Medical Care in Diabetes. (Annual publication, highly authoritative for clinical guidelines).
18. Centers for Disease Control and Prevention (CDC). (n.d.). National Diabetes Statistics Report. Retrieved from <https://www.cdc.gov/diabetes/data/statistics-report/index.html>
19. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30.
20. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Appendix

```
[1]: import os
import findspark

#  Correct your paths here:
os.environ["JAVA_HOME"] = r"C:\Program Files\Eclipse Adoptium\jdk-11.0.27.6-hotspot"
os.environ["SPARK_HOME"] = r"C:\spark-3.5.6-bin-hadoop3"

findspark.init()

[2]: from pyspark.sql import SparkSession

[3]: spark = SparkSession.builder.appName("Healthanalysis").getOrCreate()

[4]: data_path = "C:/Users/samin/Desktop/bigdata/data.csv"
df = spark.read.csv(data_path, header=True, inferSchema=True)
df.show(5)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Diabetes_012|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcCost|GenHlth|MentHlth|PhysHlth|DiffWalk|Sex| Age|Education|Income|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|          0.0|      1.0|      1.0|      1.0|40.0|      1.0|      0.0|              0.0|          0.0|      0.0|      1.0|              0.0|          1.0| |
|          0.0|      5.0|     18.0|     15.0|      1.0|0.0|      9.0|              4.0|      3.0|              0.0|      0.0|      0.0|              0.0|          0.0|
|          0.0|      0.0|      0.0|      0.0|      0.0|25.0|      1.0|              0.0|          0.0|      1.0|      0.0|      0.0|              0.0|          0.0|
|          1.0|      3.0|      0.0|      0.0|      0.0|0.0|      7.0|              6.0|      1.0|              0.0|      0.0|      0.0|              0.0|          0.0|
|          0.0|      0.0|      1.0|      1.0|      1.0|28.0|      0.0|              0.0|          0.0|      0.0|      1.0|      0.0|              0.0|          1.0|
|          1.0|      5.0|     30.0|     30.0|      1.0|0.0|      9.0|              4.0|      8.0|              0.0|      1.0|      1.0|              0.0|          1.0|
|          0.0|      0.0|      1.0|      0.0|      1.0|27.0|      0.0|              0.0|          0.0|      1.0|      1.0|      1.0|              0.0|          1.0|
|          0.0|      2.0|      0.0|      0.0|      0.0|0.0|     11.0|              3.0|      6.0|              0.0|      1.0|      1.0|              0.0|          1.0|
|          0.0|      0.0|      1.0|      1.0|      1.0|24.0|      0.0|              0.0|          0.0|      1.0|      1.0|      1.0|              0.0|          1.0|
|          0.0|      2.0|      3.0|      0.0|      0.0|0.0|     11.0|              5.0|      4.0|              0.0|      1.0|      1.0|              0.0|          1.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

[5]: df_clean = df.dropna()

[6]: df

DataFrame[Diabetes_012: double, HighBP: double, HighChol: double, CholCheck: double, BMI: double, Smoker: double, Stroke: double, HeartDiseaseorAttack: double, PhysActivity: double, Fruits: double, Veggies: double, HvyAlcoholConsump: double, AnyHealthcare: double, NoDocbcCost: double, GenHlth: double, MentHlth: double, PhysHlth: double, DiffWalk: double, Sex: double, Age: double, Education: double, Income: double]

[7]: df.columns
```

```
[10]: df.createOrReplaceTempView("health")
```

```
[11]: avg_bmi_income_by_education = spark.sql("""
      SELECT
        Education,
        ROUND(AVG(BMI), 2) AS AvgBMI,
        ROUND(AVG(Income), 2) AS AvgIncome
      FROM health
      GROUP BY Education
      ORDER BY AvgBMI DESC
    """)
avg_bmi_income_by_education.show()
```

```
+-----+-----+
|Education|AvgBMI|AvgIncome|
+-----+-----+
| 1.0| 29.76| 3.77|
| 3.0| 29.64| 3.77|
| 2.0| 29.45| 3.29|
| 4.0| 29.04| 5.19|
| 5.0| 28.88| 5.88|
| 6.0| 27.52| 6.98|
+-----+-----+
```

```
[12]: stroke_by_age = spark.sql("""
      SELECT
        Age,
        COUNT(*) AS Count
      FROM health
      WHERE Stroke = 1
      GROUP BY Age
      ORDER BY Age
    """)
stroke_by_age.show()
```

```
+-----+
| Age|Count|
+-----+
| 1.0| 21|
| 2.0| 29|
| 3.0| 83|
| 4.0| 137|
| 5.0| 229|
| 6.0| 368|
| 7.0| 722|
| 8.0| 1085|
+-----+
```



```
[13]: from pyspark.sql.functions import col, isnull, when, count
```

```
[14]: df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Diabetes_012|HighBP|HighChol|CholCheck|BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcCo|
st|GenHlth|MentHlth|PhysHlth|DiffWalk|Sex|Age|Education|Income|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
[15]: df_with_label = df.withColumn("label", when(df["Diabetes_012"] == 2.0, 1.0).otherwise(0.0))
```

```
[16]: df_with_label.createOrReplaceTempView("health")
```

```
[17]: correlation = df.stat.corr("CholCheck", "HvyAlcoholConsump")
print("Correlation between CholCheck and heavyAlcohol consumption :", correlation)
```

Correlation between CholCheck and heavyAlcohol consumption : -0.023730090654813747

```
[18]: high_risk_groups = spark.sql("""
```

```
SELECT
    HighBP,
    Smoker,
    ROUND(AVG(label), 3) AS AvgRisk,
    COUNT(*) AS People
FROM health
GROUP BY HighBP, Smoker
ORDER BY AvgRisk DESC
""")
high_risk_groups.show()
```

```
+-----+-----+-----+-----+
|HighBP|Smoker|AvgRisk|People|
+-----+-----+-----+-----+
| 1.0| 1.0| 0.257| 54279|
| 1.0| 0.0| 0.232| 54550|
| 0.0| 1.0| 0.075| 58144|
| 0.0| 0.0| 0.05| 86707|
+-----+-----+-----+-----+
```

```
[19]: df = df.na.fill({'Fruits':0, 'DiffWalk': 0})
```

```
[20]: df.show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Diabetes_012|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcC
ost|GenHlth|MentHlth|PhysHlth|DiffWalk|Sex| Age|Education|Income|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|          0.0| 1.0| 1.0| 1.0|40.0| 1.0| 0.0|          0.0|          0.0| 0.0| 1.0|          0.0| 1.0|
0.0| 5.0| 18.0| 15.0| 1.0|0.0| 9.0| 4.0| 3.0|          0.0|          1.0| 0.0| 0.0|          0.0| 0.0|
|          0.0| 0.0| 0.0| 0.0| 0.0|25.0| 1.0| 0.0|          0.0|          1.0| 0.0| 0.0|          0.0| 0.0|
1.0| 3.0| 0.0| 0.0| 0.0|0.0| 7.0| 6.0| 1.0|          0.0|          0.0| 1.0| 0.0|          0.0| 1.0|
|          0.0| 1.0| 1.0| 1.0|28.0| 0.0| 0.0|          0.0|          0.0| 1.0| 0.0| 0.0| 0.0|          0.0| 1.0|
1.0| 5.0| 30.0| 30.0| 1.0|0.0| 9.0| 4.0| 8.0|          0.0|          1.0| 1.0| 1.0|          0.0| 1.0|
|          0.0| 1.0| 0.0| 1.0|27.0| 0.0| 0.0|          0.0|          1.0| 1.0| 1.0|          0.0| 1.0|
0.0| 2.0| 0.0| 0.0| 0.0|0.0|11.0| 3.0| 6.0|          0.0|          1.0| 1.0| 1.0|          0.0| 1.0|
|          0.0| 1.0| 1.0| 1.0|24.0| 0.0| 0.0|          0.0|          1.0| 1.0| 1.0|          0.0| 1.0|
0.0| 2.0| 3.0| 0.0| 0.0|0.0|11.0| 5.0| 4.0|          0.0|          1.0| 1.0| 1.0|          0.0| 1.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
[21]: from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
[22]: df = df.drop("Sex_indexed")
indexer = StringIndexer(inputCol="Sex", outputCol="Sex_indexed")
df = indexer.fit(df).transform(df)
df.select("Sex", "Sex_indexed").show(5)
```

```
+-----+-----+
|Sex|Sex_indexed|
+-----+-----+
|0.0|          0.0|
|0.0|          0.0|
|0.0|          0.0|
|0.0|          0.0|
|0.0|          0.0|
+-----+-----+
```

only showing top 5 rows

```
[87]: rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=100)
      rf_model = rf.fit(train_data)
```

```
[88]: predictions = rf_model.transform(test_data)
      predictions.select("prediction", "label", "probability").show(5)
```

```
+-----+-----+-----+
|prediction|label|      probability|
+-----+-----+-----+
|      0.0|  0.0|[0.90896487460461...|
|      0.0|  0.0|[0.89432397503120...|
|      0.0|  0.0|[0.90940633939503...|
|      0.0|  0.0|[0.90728765166732...|
|      0.0|  0.0|[0.89309159310247...|
+-----+-----+-----+
only showing top 5 rows
```

```
[89]: evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
      accuracy = evaluator.evaluate(predictions)
      print(f"Accuracy: {accuracy}")

Accuracy: 0.8597052784152676
```

```
[90]: evaluator_f1 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="f1")
      f1_score = evaluator_f1.evaluate(predictions)
      print(f"F1 Score: {f1_score}")

F1 Score: 0.7948683926001021
```

```
[91]: evaluator_precision = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedPrecision")
      precision = evaluator_precision.evaluate(predictions)
      print(f"Precision: {precision}")

Precision: 0.7391253522034589
```

```
[92]: evaluator_recall = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedRecall")
      recall = evaluator_recall.evaluate(predictions)
      print(f"Recall: {recall}")

Recall: 0.8597052784152676
```

```
[93]: from pyspark.ml.regression import LinearRegression
```

```
[94]: lr = LinearRegression(featuresCol="features", labelCol="label")
      lr_model = lr.fit(train_data)
```

```
print(f"Recall: {recall:.4f}")
```

```
Accuracy: 0.8670  
F1 Score: 0.8382  
Precision: 0.8389  
Recall: 0.8670
```

```
[109]: df.groupby("label").count().show()
```

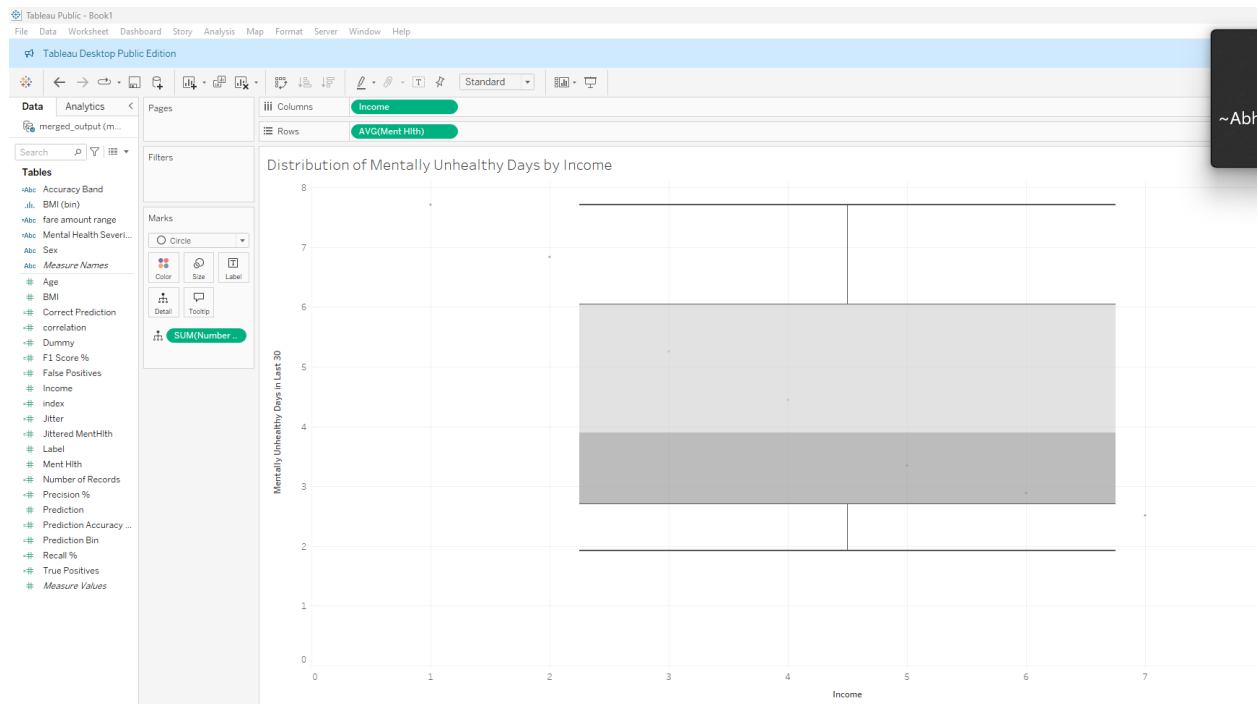
```
+-----+  
|label| count|  
+-----+  
| 0.0|213703|  
| 1.0| 35346|  
+-----+
```

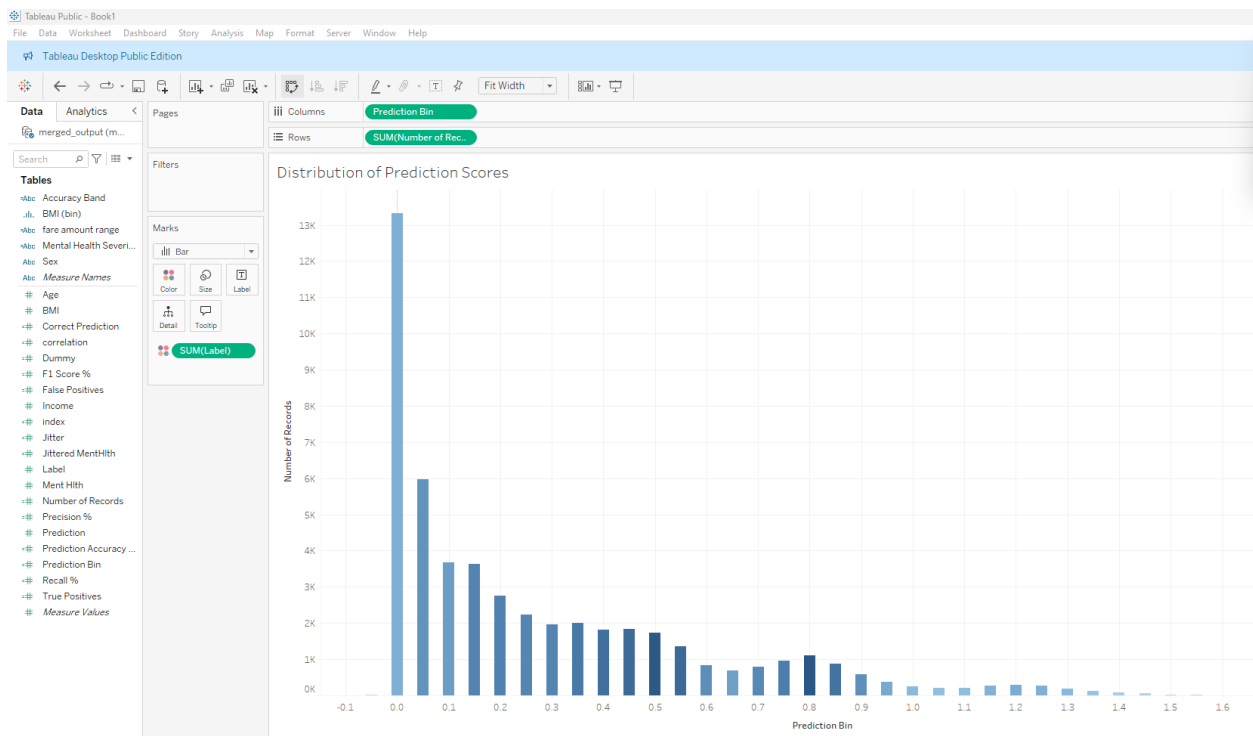
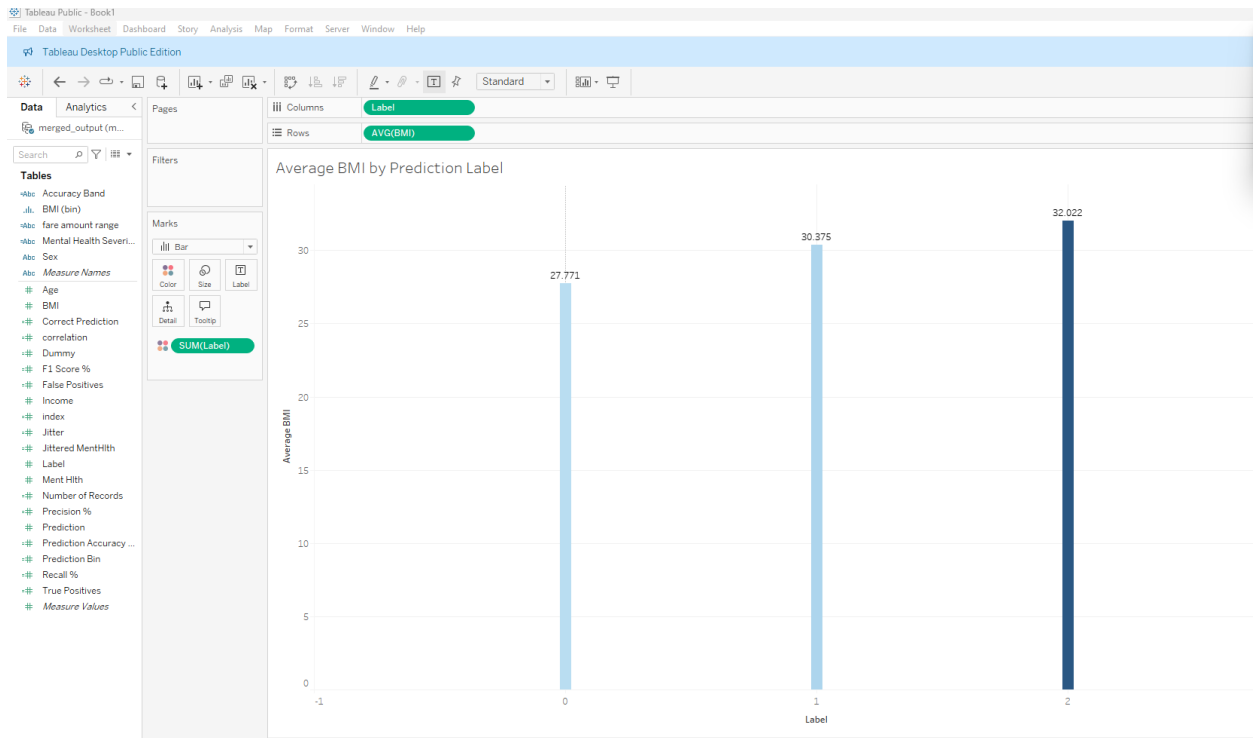
```
[107]: predictions.show(5)
```

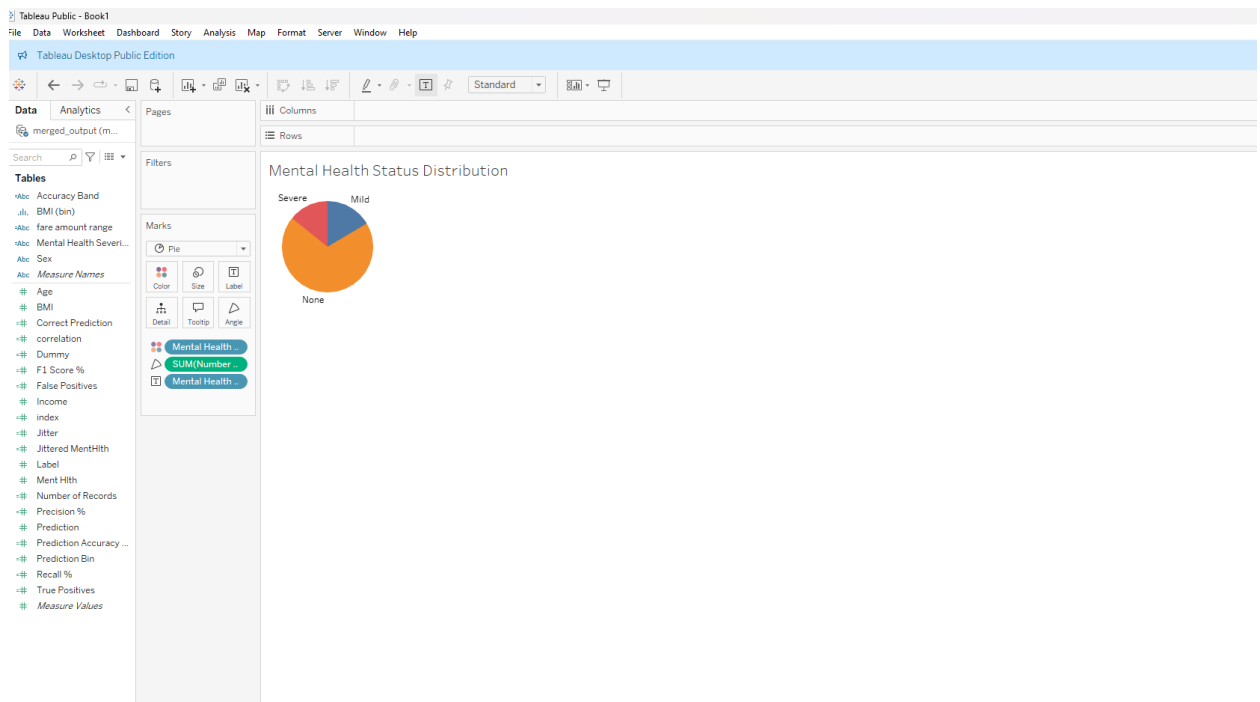
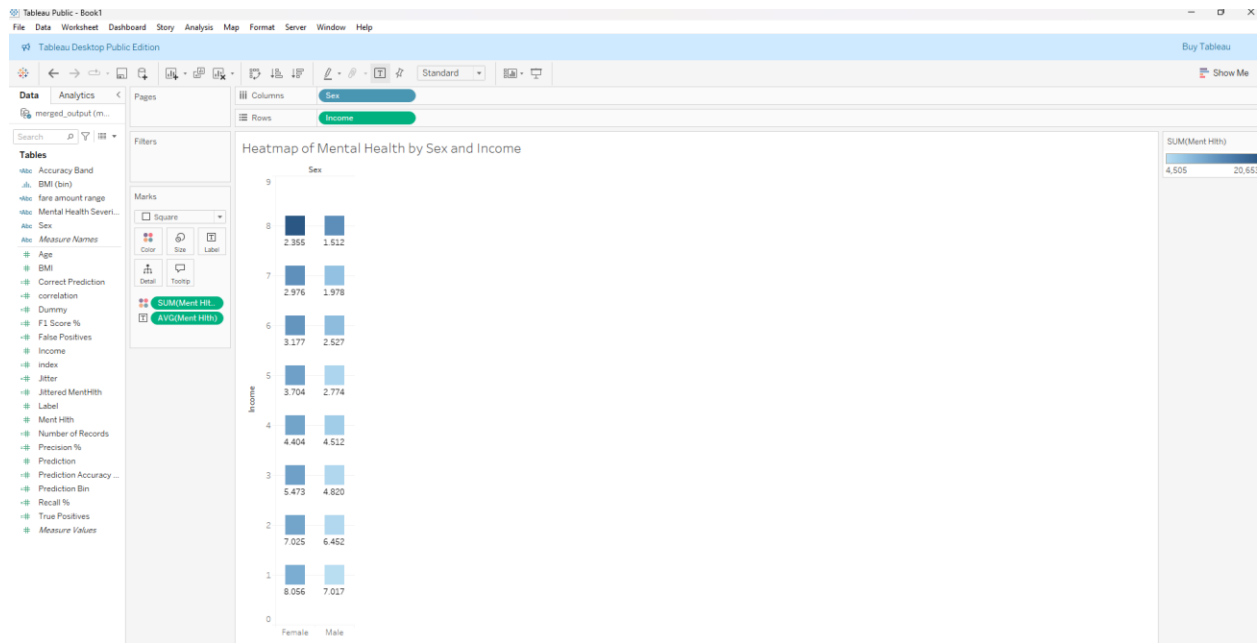
```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|label|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|HeartDiseaseorAttack|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthcare|NoDocbcCost|Gen  
Hlth|MenthHlth|PhysHlth|DiffWalk|Sex| Age|Education|Income|Sex_indexed| Sex_encoded| features| rawPrediction| probability|p  
rediction|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 0.0| 0.0| 0.0| 0.0|16.0| 1.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0|  
2.0| 5.0| 0.0| 0.0|1.0| 5.0| 6.0| 8.0| 1.0| (1,[],[])|(21,[3,4,7,11,13,...|[2.09645083812110...|[0.98512229014955...|  
0.0|  
| 0.0| 0.0| 0.0| 0.0|17.0| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 1.0| 0.0| 0.0| 1.0| 0.0|  
3.0| 0.0| 0.0| 0.0|1.0| 9.0| 6.0| 1.0| 1.0| (1,[],[])|(21,[3,7,9,11,13,...|[1.84032952094669...|[0.97541338176971...|  
0.0|  
| 0.0| 0.0| 0.0| 0.0|17.0| 0.0| 0.0| 0.0| 0.0| 1.0| 1.0| 1.0| 0.0| 1.0| 0.0|  
1.0| 0.0| 0.0| 0.0|0.0| 4.0| 6.0| 8.0| 0.0|(1,[0],[1.0])|(21,[3,7,8,9,11,1...|[2.17723075061643...|[0.98731365430154...|  
0.0|  
| 0.0| 0.0| 0.0| 0.0|17.0| 1.0| 0.0| 0.0| 0.0| 1.0| 1.0| 1.0| 0.0| 0.0| 1.0|  
2.0| 5.0| 0.0| 0.0|0.0| 7.0| 4.0| 3.0| 0.0|(1,[0],[1.0])|(21,[3,4,7,8,9,12...|[1.95411170073387...|[0.98031898217086...|  
0.0|  
| 0.0| 0.0| 0.0| 0.0|18.0| 0.0| 0.0| 0.0| 0.0| 1.0| 1.0| 1.0| 0.0| 1.0| 0.0|  
3.0| 0.0| 0.0| 0.0|1.0|10.0| 2.0| 4.0| 1.0| (1,[],[])|(21,[3,7,8,9,11,1...|[1.68812060530150...|[0.96695370654357...|  
0.0|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

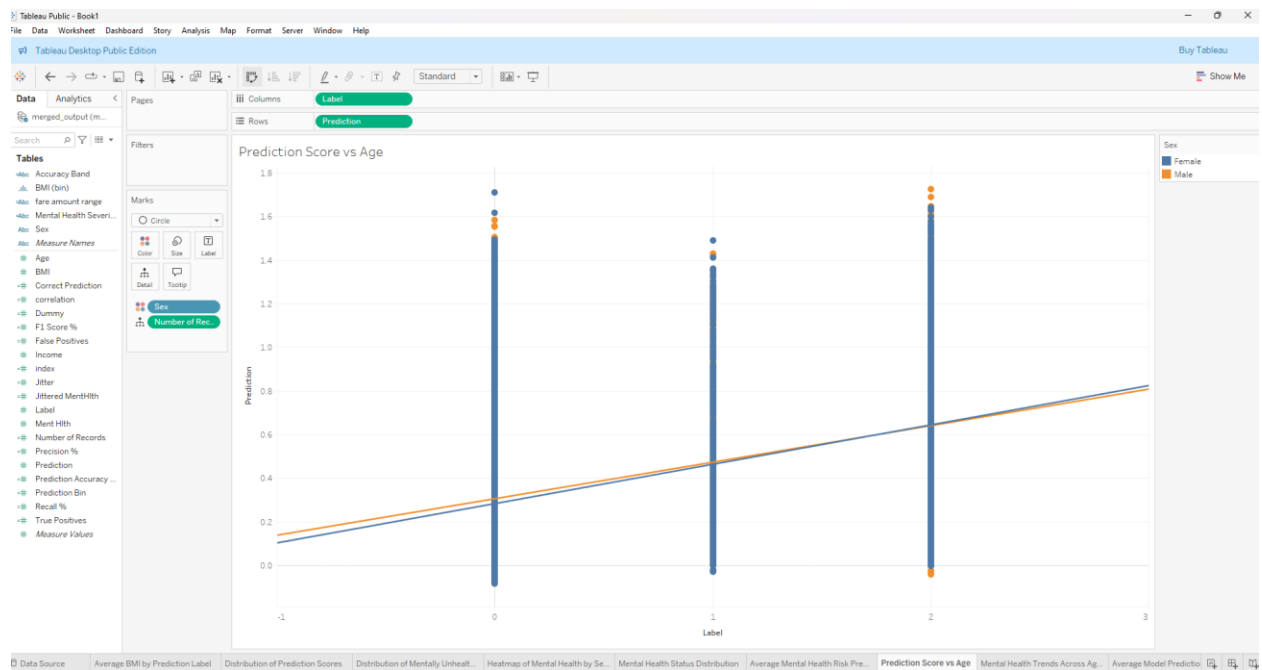
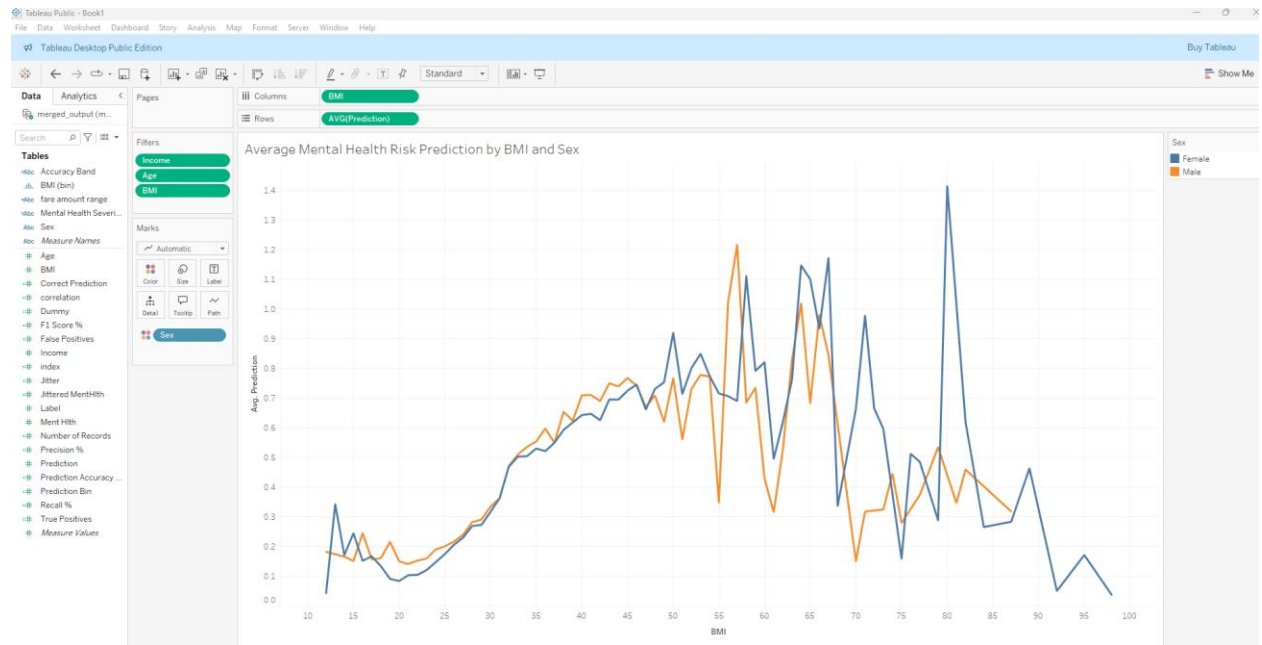
only showing top 5 rows

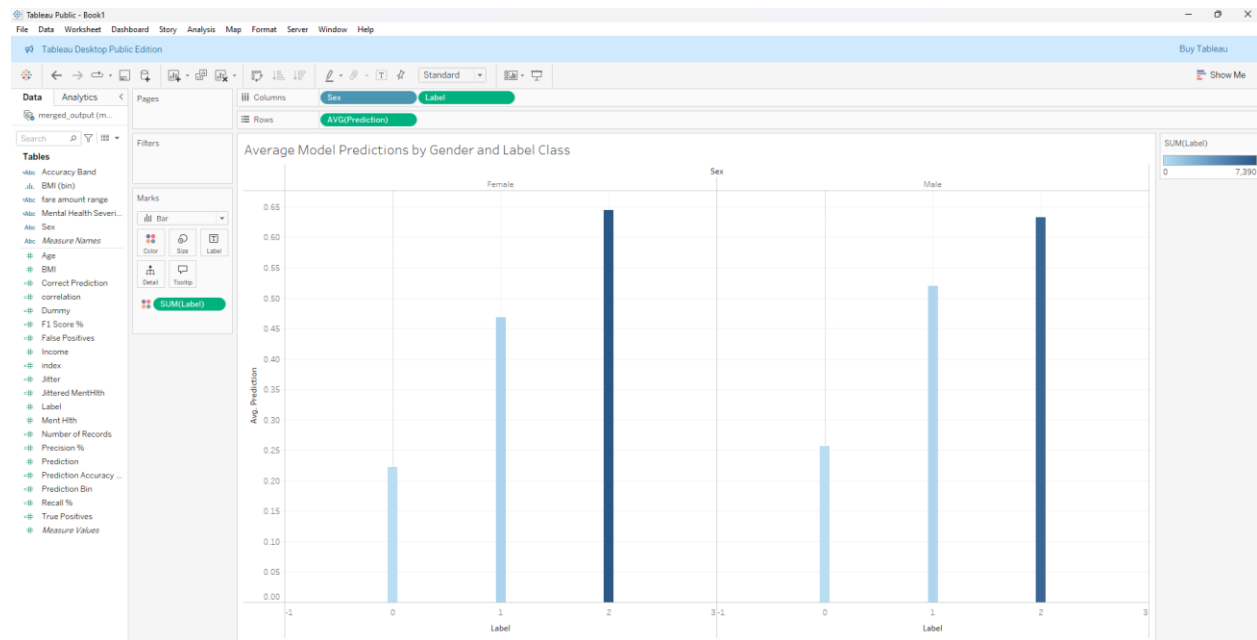
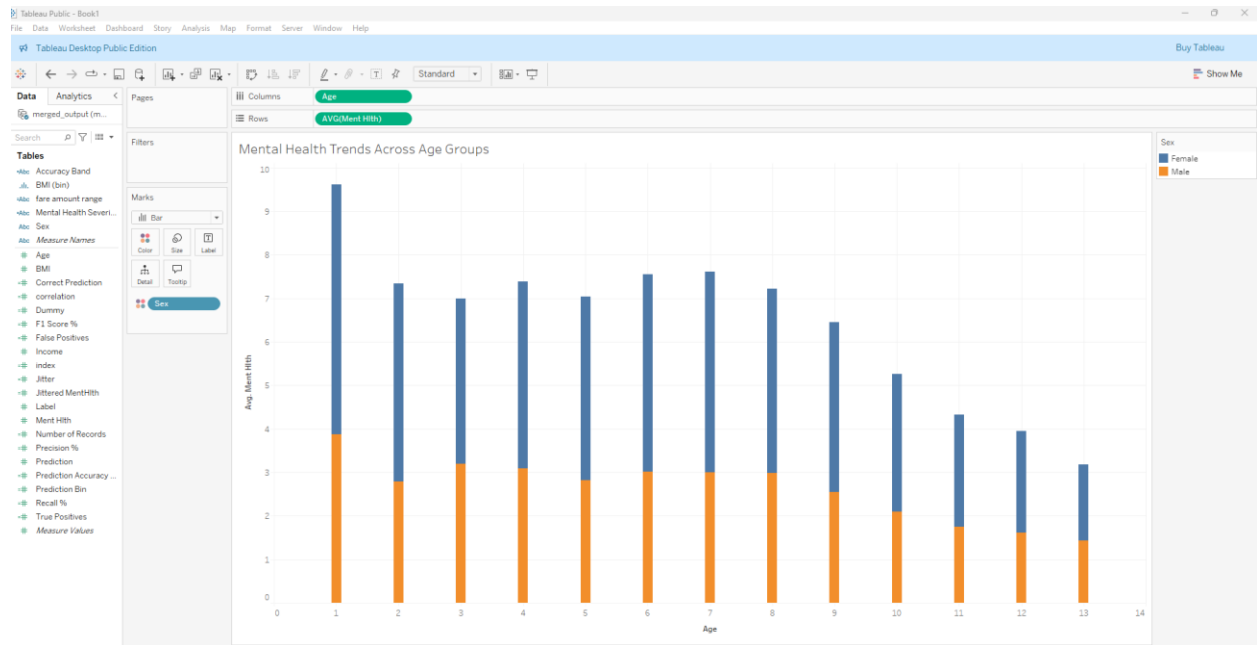
```
[108]: output_df = predictions.select("label", "prediction", "MenthHlth", "BMI", "Age", "Income")  
output_df.write.csv("diabetes_prediction.csv", header=True, mode="overwrite")
```











Link

<https://github.com/saminkc999/BigData>

<https://public.tableau.com/app/profile/samin.kc/vizzes>