Insuring Smiles: Predicting routine dental coverage using Spark ML Algorithms

Authors: Aishwarya Gupta, Rahul Sarvottam Bhogale, Priyanka Thota, Prathushkumar Dathuri
Department of Information Systems, California State University Los Angeles
CIS5560 System Analysis and Design
agupta25@calstatela.edu, rbhogal@calstatela.edu, pthota2@calstatela.edu, pdathur@calstatela.edu

Abstract: Finding suitable health insurance coverage can be challenging for individuals and small enterprises in the USA. The Health Insurance Exchange Public Use Files (Exchange PUFs) dataset provided by CMS offers valuable information on health and dental policies [1]. In this project, we leverage machine learning algorithms to predict if a health insurance plan covers routine dental services for adults. By analyzing plan type, region, deductibles, outofpocket maximums, and copayments, we employ Logistic Regression, Decision Tree, Random Forest, Gradient Boost, Factorization Model and Support Vector Machine algorithms. Our goal is to provide a clinical strategy for individuals and families to select the most suitable insurance plan based on income and expenses

1. Introduction

Our work focuses on using machine learning algorithms to predict dental coverage in health insurance plans. We chose this topic due to the challenge individuals and small enterprises face in finding suitable health insurance coverage. By leveraging the CMS dataset [1], we aim to provide insights on routine dental service coverage based on plan details and other relevant factors.

he importance of our work lies in supporting timely benefits analysis, rate evaluation, copayment assessment, premium analysis, and geographic coverage analysis. By applying machine learning algorithms such as Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, Gradient Boost and Factorization Machine we contribute to the field by offering practical recommendations for selecting appropriate health insurance plans based on income and expenses.

Our background involves utilizing big data technologies and advanced machine learning techniques to improve classification accuracy. By comparing the performance of different algorithms, we provide valuable insights for individuals and insurance stakeholders in making informed decisions regarding health coverage options.

2. Related Work

1. In one of the similar works found in the Hamilton Project, smart policies on health insurance were developed using the Healthcare.gov dataset [1]. The study aimed to create a platform for suggesting smart policies based on cost reductions and insurance coverage offered by employers. In contrast, our analysis focuses on predicting dental coverage in health insurance plans and determining if routine dental services are included. Additionally, our approach involves utilizing big data file management, HDFS, along with PySpark ML using Databricks/Zeppelin notebooks.

2. Another related work was presented by the NAIC (National Association of Insurance Commissioner) [3]. Their study primarily focuses on the business side of Health insurance marketplace data and analyzes the growth in
issuing plans over a 10year period. They observe a decrease in net earnings% and profit margin%. In contrast, our work focuses on providing insights and predictions related to routine health and dental checkup services, with a focus on consumer needs.

3. The work presented by the AMA (American Medical Association) emphasizes health insurance companies with wider market coverage and explores metrics such as enrollment% and small groups% [4]. In contrast, our analysis differs in terms of deliverables as we concentrate on individual/family plan needs. We compare various machine learning models to gain insights into different insurance plans, specifically regarding dental coverage and routine services.

The key difference in our work lies in the utilization of big data technologies such as Hadoop, PySpark ML, and cloud computing. Our analysis leverages the power of big data processing and analytics to predict dental coverage in health insurance plans, offering valuable insights for individuals and small enterprises in selecting the most suitable insurance options.

3. Background Work

Similar work found in this paper where it analyzes health insurance coverage and utilization of health services in 38 Sub-Saharan African countries using recent Demographic Health Survey (DHS) data. The study reveals low enrolment rates and utilization of health services in the region. Machine learning (ML) techniques significantly improve the accuracy of predicting health insurance coverage compared to traditional approaches. The findings underscore the need to identify and include the excluded population and highlight the importance of country specific factors in predicting enrolment. The research demonstrates the potential of ML models to target policies and improve overall targeting of the excluded population. Efforts to achieve Universal Health Coverage (UHC) should focus on improving coverage and utilization rates. [5]

4. Specification

The dataset comprises of insurance plans related to planning rates, types, coverage, rules, family members, etc. The dataset is of size 2.1 GB having a duration from 2017 to 2021 [1]. Table 1 shows numerous files, file types, and their size of it.

Table 1 Data Specification

Data Set	Size (Total 2.1 GB)
benefits_cost_sharing.csv	2,050,850 KB
benefits1.csv (Sampled dataset)	12,901 KB

6.

The below table shows the hardware specifications for the Hadoop cluster and CPU configuration

Table 2 H/W Specification

Hadoop Specs	
Master Nodes	2
Worker Nodes	3
Version	3.1.2
Storage	390.7 GB
CPU Configuration	
CPU core	6
Frequency	1995.312z

5. Implementation Flowchart

The outlined flowchart illustrates the process of analyzing a dataset obtained from CMS gov website, beginning with acquiring the dataset and uploading it to a Linux system, storing it in HDFS. The subsequent steps involve loading the dataset, performing initial data exploration, handling missing values, as well as removing irrelevant columns. If necessary, data transformation and normalization are applied, followed by data quality checks. A modeling pipeline is defined, encompassing tasks such as data splitting, feature engineering, data scaling, and encoding categorical variables. Models are built by selecting algorithms, training them, and saving the trained models. Evaluation metrics are used to compare algorithms for their accuracy and performance. Prediction is applied to the testing set, evaluating performance, and comparing their effectiveness. The flowchart serves as a guide for a structured and comprehensive approach to the data analysis process.(Figure 1)

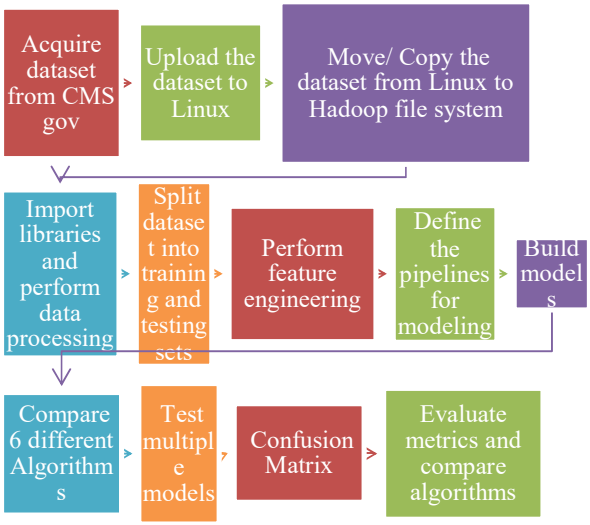

Figure 1 Implementation Flowchart

6. Data Preprocessing

As part of the data preparation process before building the pipeline, we needed to convert categorical data into numerical data since the dataset contained nonnumeric values. Since the dataset was balanced, with 60% of health insurance plans covering dental services and the remaining 40% not covering dental services, we opted to use classification algorithms.

For testing purposes, we randomly sampled only 50,000 rows in by using Python code in a Jupyter notebook (Figure 2) and later loaded into DataBricks (Figure 3). We selected independent columns for our prediction, including BusinessYear, StateCode, IssuerId, SourceName, IsEHB, QuantLimitOnSvc, Exclusions, EHBVarReason, and IsCovered, to ensure unbiased predictions.

Next, we identified and removed null values only from the prediction column (Figure 4). The remaining null values in other columns were replaced with the maximum value from each respective column (Figure 5). Later convert the 'IsCovered' column into 0 and 1 for classification modelling as this is our prediction column (Figure 6).

After preprocessing the dataset, we split it into a 70% training set and a 30% testing set for modeling purposes.

```python
import pandas as pd
import random

# Load the original dataset file into a DataFrame
df = pd.read_csv('C:\\Users\\agupta25\\Desktop\\5560 project\\benefits_cost_sharing.csv')

# Randomly select 10,000 rows from the DataFrame
num_rows = min(50000, len(df))  # Select either 50,000 rows or the total number of rows if it's less than 50,000
random.seed(42)  # Set a seed for reproducibility
indices = random.sample(range(len(df)), num_rows)
df_sample = df.loc[indices]

# Save the selected rows as a new CSV file
df_sample.to_csv('benefits1.csv', index=False)
df_sample
```

Figure 2 – Sampled dataset using Python

```
###Load the sample dataset from DBFS

# Oracle BDCE
#csv = spark.read.csv('/user/agupta25/project/benefits1.csv', inferSchema=True, header=True)
# File location and type
file_location = "/FileStore/tables/benefits1.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

Figure 3 – Loaded sampled csv in Databricks

```
###Replace null or whitespace values with None. Later drop the values.

from pyspark.sql.functions import when, col

# Replace empty strings or whitespace with null values
df = df.withColumn('label', when(col('label').isin('', ' '), None).otherwise(col('label')))

# Drop null values from label column
df = df.dropna(subset=['label'])
df.show()
```

Figure 4 – Drop the null value of prediction column

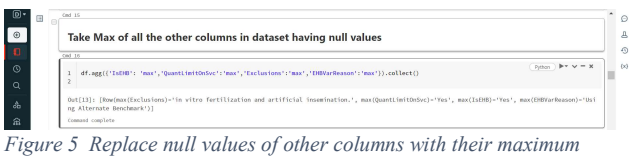

Figure 5 Replace null values of other columns with their maximum

Figure 6 Labelled into 0's and 1's

7. Defining Pipeline

- A pipeline consists of a series of transformer and estimator stages that typically prepare a DataFrame for
- modeling and then train a predictive model. In this case, you will create a pipeline with seven stages:
- A StringIndexer estimator that converts string values to indexes for categorical features
- A VectorAssembler that combines categorical features into a single vector
- A VectorIndexer that creates indexes for a vector of categorical features
- A VectorAssembler that creates a vector of continuous numeric features
- A MinMaxScaler that normalizes continuous numeric features
- A VectorAssembler that creates a vector of categorical and continuous features

A predictive model often requires multiple stages of feature preparation. For example, it is common when using some algorithms to distinguish between continuous features (which have a calculable numeric value) and categorical features (which are numeric representations of discrete categories). It is also common to normalize continuous numeric features to use a common scale (for example, by scaling all numbers to a proportional decimal value between 0 and 1) (Figure 7).

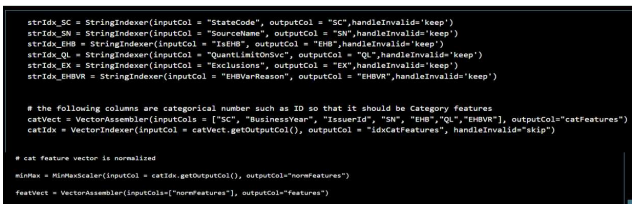

Figure 7 – Defining Pipeline

8.1 Training

In our analysis, we utilize six ML algorithms to predict dental coverage in health insurance plans.
These algorithms are Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Factorization Machine (FM), Gradient Boost (GBT), and Support Vector Machine (SVM). (Figure 8)
To optimize their performance, we tune the hyperparameters using a parameter grid (Figure 9).
For evaluation, we employ a Cross-Validator model that compares the predicted values with the true labels (IsCovered column) using a binary classification evaluator (Figure 10).
Metrics such as accuracy, precision, and recall are utilized to assess the performance of these algorithms.

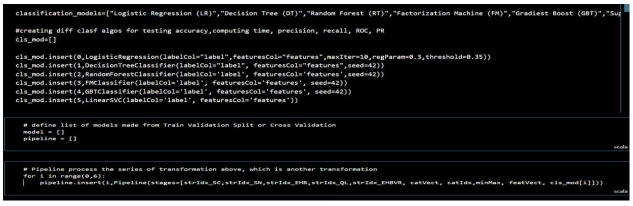

Figure 8 – Defining Models

```
paramGrid.insert(5,ParamGridBuilder() \
             .addGrid(cls_mod[5].regParam, [0.01, 0.5]) \
             .addGrid(cls_mod[5].maxIter, [1, 5]) \
             .addGrid(cls_mod[5].tol, [1e-4, 1e-3]) \
             .addGrid(cls_mod[5].fitIntercept, [True, False]) \
             .addGrid(cls_mod[5].standardization, [True, False]) \
             .build())
```

Figure 9 – Parameter building

Figure 10 – Training Model

8.2 Testing

The trained model, which consists of all the stages in the pipeline, is a transformer that can be applied to a specific DataFrame to generate predictions. This model is used to verify the predictions on the test dataset, which represents 30% of the actual dataset. The resulting DataFrame contains the predicted values in the prediction column and the actual known values in the trueLabel column. (Figure 11.1 & 11.2)

Figure 11.1 – Testing Model

features	prediction	trueLabel
[1.0,0.0,0.709384...]	1.0	1
(7,[0,2,4],[1.0,0...]	1.0	1
[0.894736842105263...]	1.0	1
[0.894736842105263...]	1.0	1
[0.894736842105263...]	1.0	1
[0.894736842105263...]	1.0	1
[0.894736842105263...]	1.0	0
[0.657894736842105...]	1.0	1
[0.657894736842105...]	1.0	1
[0.657894736842105...]	1.0	1
[0.657894736842105...]	1.0	1
[0.657894736842105...]	1.0	1
[0.657894736842105...]	1.0	1
[0.657894736842105...]	1.0	0
[0.657894736842105...]	1.0	0
[0.210526315789473...]	1.0	1
(7,[0,2,4],[0.210...]	1.0	1
(7,[0,2,4],[0.210...]	1.0	1

Figure 11.2 – Prediction and true label on test dataset

9. Evaluation & Prediction

The evaluation metrics used for performance assessment of the ML algorithms are as follows:

Computing time: Time required for computations and running ML algorithms on the dataset. (Figure 12)

Confusion matrix: Matrix indicating true positives, true negatives, false positives, and false negatives.
Precision: Proportion of correctly predicted positive instances out of the total instances predicted as positive.

Recall: Proportion of correctly predicted positive instances out of all actual positive instances. (Figure 13)

ROC (Receiver Operating Characteristic): Graphical representation of the model's performance by plotting true positive rate against false positive rate.
PR (Precision-Recall): Graphical representation of precision against recall at different classification thresholds.
Accuracy: Ratio of correctly classified instances to the total number of instances.
F1 Score: Harmonic mean of precision and recall, providing a balanced measure of the model's performance.
These metrics help assess the accuracy, sensitivity, specificity, and overall correctness of the models' predictions. (Figure 14)

Figure 12 – Computing time

metric	value
TP	11699.0
FP	2713.0
TN	0.0
FN	3.0
Precision	0.8117540938107133
Recall	0.9997436335669116

Figure 13 – Confusion Matrix

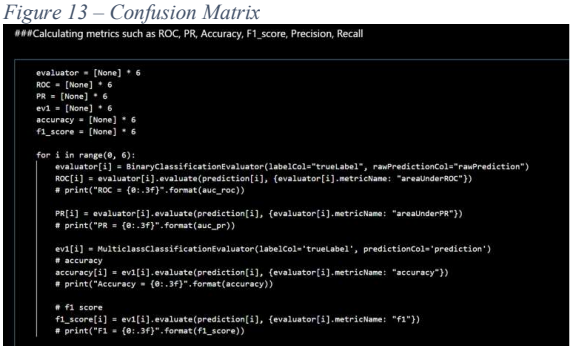

Figure 14 – Binary & Multi Classification Evaluator

10. PySpark CLI

Export the Databricks file with .py extension and run the code in CLI using the command and could see from Figure 15.

```
$ spark-submit filename.py
```

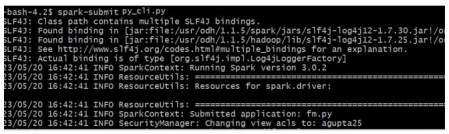

Figure 16.2 – Actual dataset summary

11. Conclusion

1. From the above tables, we come to conclusion that GBT is the best model with highest precision of 85% with least computation time 6.5 mins for the **sampled dataset** (Figure 16.1)

Models	Computation Time	Precision	Recall
Logistic Regression (LR)	5 mins	81%	0.99
Decision Tree (DT)	30 mins	87%	0.99
Random Forrest (RF)	36 mins	86%	0.99
Factorization Machine (FM)	13 mins	81%	0.99
Gradient Boost (GBT)	6.5 mins	85%	0.99
Support Vector Machine (SVM)	15.5 mins	81%	1.00

2. The best model is Gradient Boost with computation time 40 minutes and accuracy with 85% with **actual dataset** (Figure 16.2)

Models	Computation Time	Precision	Recall
Logistic Regression (LR)	20.5 mins	0.81	0.99
Gradient Boost (GBT)	40 mins	0.85	0.99
Factorization Machine (FM)	5 mins	0.81	0.99
Support Vector Machine (SVM)	4 mins	0.81	1.00

3. Feature Importance is the ranking of df columns and observing which columns are impacting the most for predicting whether the dental services are covered or not in health insurance plans based on the value. Thus, we evaluated only for **GBT Classifier**.

4. ***Exclusion column is having the highest impact with 0.55*** compared to isEHB which is redundant column.

Feature Importance:

Ranking	Df columns	Importance value
1	Exclusions	0.555733
2	BusinessYear	0.162801
3	IssuerId	0.131934
4	QuantLimitOnSvc	0.12204
5	SourceName	0.015729
6	StateCode	0.011764
7	IsEHB	0

Model	Logistic Regression (LR)	Decision Tree (DT)	\
Computation Time (min)	5.152021	29.827601	
ROC	0.620981	0.648268	
PR	0.879197	0.847665	
Accuracy	0.727362	0.845612	
F1 Score	0.727362	0.845612	
Precision	0.811754	0.866900	
Recall	0.999744	0.993505	

Model	Random Forest (RT)	Factorization Machine (FM)	\
Computation Time (min)	36.103577	13.262092	
ROC	0.846004	0.633083	
PR	0.958525	0.882931	
Accuracy	0.834552	0.727156	
F1 Score	0.834552	0.727156	
Precision	0.858991	0.811676	
Recall	0.998462	0.999231	

Model	Gradiest Boost (GBT)	Support Vector Machine (SVM)
Computation Time (min)	6.539898	15.529445
ROC	0.828126	0.601684
PR	0.953801	0.870743
Accuracy	0.824062	0.727465
F1 Score	0.824062	0.727465
Precision	0.852945	0.811793
Recall	0.999744	1.000000

Figure 16.1 – Evaluation Metrics

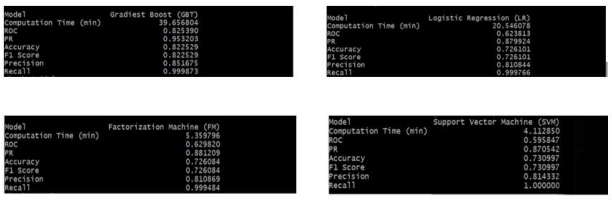

Figure 16.2 – Evaluation Metrics

12. References

[1] Dataset – Centers for Medicare & Medicaid Services (CMS)
https://www.cms.gov/cciio/resources/dataresources/marketplacepuf

[2] Ben Handel; Jonathan KolstadSmart policies on Health Insurance choice, University of California, Berkely,2015
https://www.hamiltonproject.org/assets/files/smart_policies_on_health_insurance_choice_final_proposal.pdf

[3] National Association of Insurance Commisioners(NAIC) 2021 Annual Results, USA, 2021
https://content.naic.org/sites/default/files/2021AnnualHealthInsuranceIndustryAnalysisReport.pdf

[4] American Medical Association(AMA)Competition Health Insurance US markets, Chicago, 2021

https://www.amaassn.org/system/files/competitionhealth insuranceusmarkets.pdf

[5] Development Economics Group & Center for Development and Cooperation (NADEL), ETH Zurich, Clausiusstrasse 37, 8092 Zurich, Switzerland
https://www.un.org/ldc5/sites/www.un.org.ldc5/files/durizzoguentherharttgen_ldc_future_forum.pdf

[6] GitHub – https://github.com/rahulbhogale

[7] PySpark documentation:
https://spark.apache.org/docs/latest/api/python/reference/pyspark.ml.html#classification

[8] GitBash: https://gitforwindows.org/

[9] Databricks Community Edition:
https://community.cloud.databricks.com/login.html

[10] Jupyter Notebooks: https://jupyter.org/

[11] Source codes:
https://github.com/rahulbhogale/5560/tree/main/codes