a. 5 data processing and analysis issues related to healthcare applications:

1.Data storage and management of large volumes of COVID-19 patient records. Special user information may be heterogeneous data (such as relational database data, image data, and data collected by other apps, such as location data, some data storage maybe huge)

2.Integration of data from various sources.

3.Data privacy and security.

4.Real-time processing and analysis of COVID-19 patient records.

5.Predictive analysis using machine learning algorithms/deep learning algorithms. Algorithm tuning difficulty and improvement

b. A proposed big data technology and cloud-based application for the case study could have the following components:

Data ingestion and storage – to collect and store COVID-19 patient records in real-time.
We can use Hadoop clusters stores massive data, maybe we setup clusters on AWS or Azure.
Data processing and analysis – to process and analyze the collected data using machine learning algorithms. Spark for real-time data processing and analysis.
Data Collection: Collect the relevant COVID-19 patient data, such as patient demographics, medical history, test results, in our case we get data from hadoop clusters or other kafka tunnels or api to gather related data.
Data Cleaning: Clean the collected data by removing missing values, inconsistent values, and outliers to ensure accuracy and consistency.
Data Preprocessing: Preprocess the data to prepare it for analysis. This may include normalization, scaling, or encoding of categorical variables.
Data Exploration: Explore the data to identify patterns and relationships between variables, such as correlations between symptoms and test results.
Feature Engineering: Create new features by combining or transforming existing variables that could improve the accuracy of predictive models.
Model Building: Train machine learning models using the preprocessed data. This may include decision trees, random forests, support vector machines, etc.
Model Validation: Validate the models using cross-validation techniques to determine their accuracy and robustness.
Model Deployment: Deploy the models to a production environment to make predictions on new patient records.
Model Monitoring and Maintenance: Monitor the models' performance and update them regularly to ensure they remain accurate and up-to-date.

Data visualization – to present the analyzed data in a clear and concise format.

Select appropriate visualizations: Choose the right type of visualization, such as bar charts, line charts, or scatter plots, to effectively represent the data and its relationships.
Choose appropriate tools: There are various data visualization tools available, such as **Tableau, PowerBI, Matplotlib, and Seaborn**, to create visualizations. Choose the tool that best suits your needs and skills.
Create visualizations: Use the selected tool to create visualizations based on the prepared data. This may involve selecting data fields, grouping data, and adding labels and annotations.
Interact with visualizations: Add interactivity, such as drill-downs and filters, to allow users to explore the data and its relationships in more detail. Like **d3.js/ chart.js** , we can use it here.

Predictive Machine learning modeling – to predict future trends and patterns in the COVID-19 patient records. We use all machine learning and deep learning algorithm to predict results, like MLLib on spark Machine learning. For COVID-19 case, we do following steps in general:

Data preparation: Collect, clean and prepare the COVID-19 patient records data for modeling.
Feature selection: Select the relevant features that have a significant impact on the outcome of interest.
Model selection: Choose the appropriate machine learning model, such as regression, decision trees, random forest, etc., based on the data and problem at hand.
Training: Train the selected model on a portion of the data set.
Validation: Evaluate the model's performance on a separate validation set.
Hyperparameter tuning: Fine-tune the model's hyperparameters to optimize its performance.
Model deployment: Deploy the final model to make predictions on new, unseen COVID-19 patient records data.
Model maintenance: Regularly monitor and update the model as new data becomes available to ensure its continued accuracy.
Model interpretability: Ensure that the model's predictions are transparent and understandable to stakeholders.

Data security and privacy – to ensure that the collected data is protected and confidential.To ensure data security and privacy for COVID-19 patient records, the following steps can be taken:
Encryption: Encrypting sensitive patient data to prevent unauthorized access.
Access control: Implementing strict access control measures to ensure that only authorized personnel have access to patient data.
Data backup: Regularly backing up data to protect against data loss or theft.
Data anonymization: Anonymizing sensitive patient data, such as names and addresses, to protect their identity.
Compliance with regulations: Adhering to relevant regulations, such as HIPAA in the US, to ensure that patient data is handled in a secure and privacy-compliant manner.
Regular security audits: Conducting regular security audits to identify and address any potential vulnerabilities.

Employee training: Providing employees with regular training on data security and privacy to ensure they are aware of the importance of protecting patient data

c. 5 reasons for choosing big data technology for the proposed application:

1.**Scalability**: Big data technology offers the ability to scale data processing and storage to accommodate the growing volume of COVID-19 patient records. This is especially important given the rapidly increasing number of cases and the need to track and manage the data effectively.

2.**Faster data processing and analysis**: With big data technology, data processing and analysis can be done in real-time, allowing for quick and effective decision making. This is critical in the case of COVID-19, where timely analysis can help to mitigate the spread of the disease.

3.**Improved data visualization** and reporting: Big data technology enables the creation of interactive, user-friendly dashboards and visualizations of COVID-19 patient data. This helps to present complex data in a way that is easy to understand and makes it easier to identify patterns and trends.

4.**Better data privacy and security:** Big data technology offers enhanced security and privacy features, such as encryption and access controls, which can help to protect sensitive COVID-19 patient data from unauthorized access and breaches.

5.**Increased efficiency and cost savings**: Big data technology provides cost savings and increased efficiency by streamlining data processing and analysis, reducing manual data entry and management, and automating reporting processes. This can help to reduce costs associated with COVID-19 patient record management and analysis.

d. To generate actual output evidence for the proposed big data technology and cloud-based application, a minimum of 30% of the proposed components can be implemented and tested. For data processing and analysis, big data tools such as Apache Spark or Apache Hadoop can be used to implement the framework.

answer: We collected the user's age","bmi","drink","coronary_myocarditis","diabetes
The data set is saved in the hadoop cluster, and then various data exploration, data analysis, data cleaning, data modeling, machine learning model testing and tuning are performed through spark ml, and our visual verification in 1(b):

1. **Data ingestion and storage**
2. **Data processing and analysis**
3. **Data visualization**

4. Predictive Machine learning modeling

The full code is at i2spark_ml_demo.ipynb, here are main steps screenshot:

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import HiveContext, Row
from pyspark.sql.types import IntegerType
import json
import sys
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

inputFile = 'mydata/health_data.csv'


conf = SparkConf().setAppName("SparkSQLAirTransit")
SparkConf().set("spark.sql.legacy.timeParserPolicy","LEGACY")

sc = SparkContext.getOrCreate(conf=conf)
hiveCtx = HiveContext(sc)
print("Loading data from " + inputFile)
```

```
Loading data from mydata/health_data.csv
```

```
/Users/abel/Envs/samaritan0/lib/python3.9/site-packages/pyspark/sql/context.py:718: FutureWarning: HiveContext is dep
recated in Spark 2.0.0. Please use SparkSession.builder.enableHiveSupport().getOrCreate() instead.
  warnings.warn(
/Users/abel/Envs/samaritan0/lib/python3.9/site-packages/pyspark/sql/context.py:112: FutureWarning: Deprecated in 3.0.
0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
```

Demo reading a CSV file stored in Hadoop using PySpark:

''' Create a Spark session spark = SparkSession.builder.appName("ReadFromHadoopCSV").getOrCreate()

Read the CSV file from Hadoop df = spark.read.csv("hdfs://:/", header=True, inferSchema=True)

'''

WQD7009 : BIG DATA APPLICATIONS AND ANALYTICS

[37]:
```
df = hiveCtx.read.option("header",True).csv(inputFile,inferSchema =True)
df.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- bmi: integer (nullable = true)
 |-- drink: integer (nullable = true)
 |-- coronary_myocarditis: integer (nullable = true)
 |-- diabetes: integer (nullable = true)
```

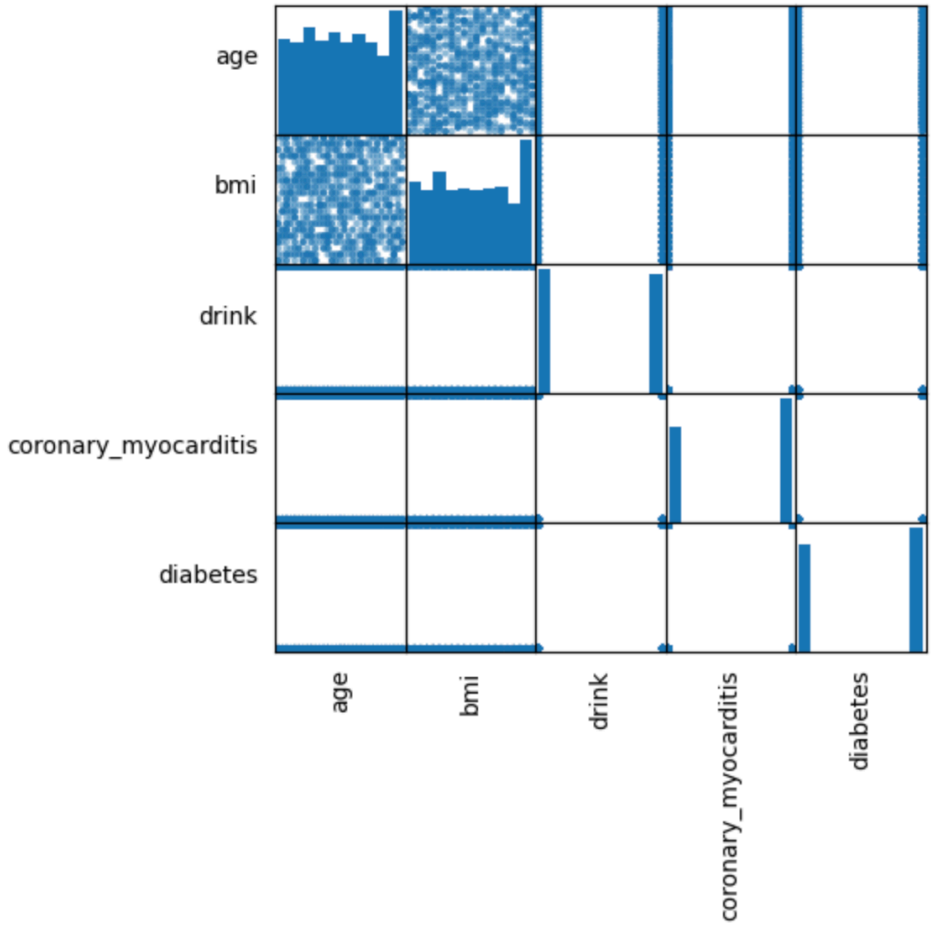
[38]:
```
print('View the data sample and perform data retrieval')
pd.DataFrame(df.take(10), columns=df.columns).transpose()
```

```
View the data sample and perform data retrieval
```

[38]:

	0	1	2	3	4	5	6	7	8	9
age	89	87	82	76	68	81	87	75	60	90
bmi	23	29	30	20	24	11	23	22	14	29
drink	0	0	0	1	1	1	0	1	0	0
coronary_myocarditis	0	1	0	1	0	1	0	1	0	1

Similarity Analysis



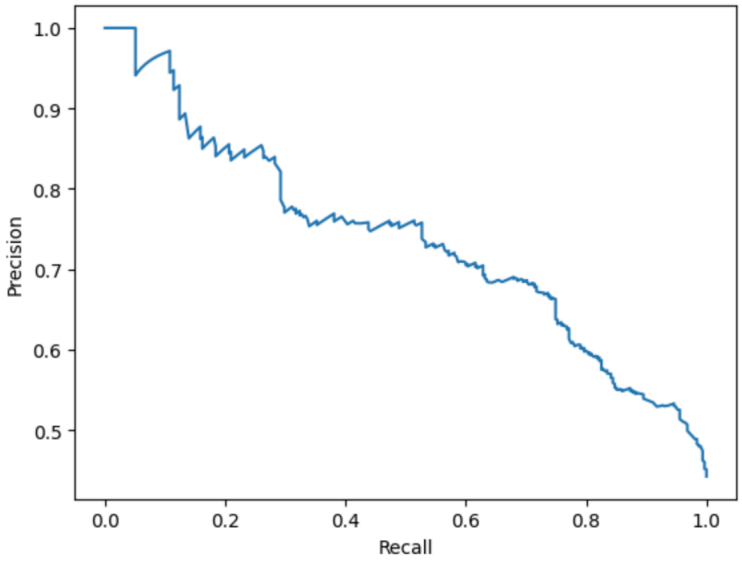

```
print('dataset set split')
train, test = df.randomSplit([0.7, 0.3], seed = 2018)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
dataset set split
Training Dataset Count: 712
Test Dataset Count: 288
```

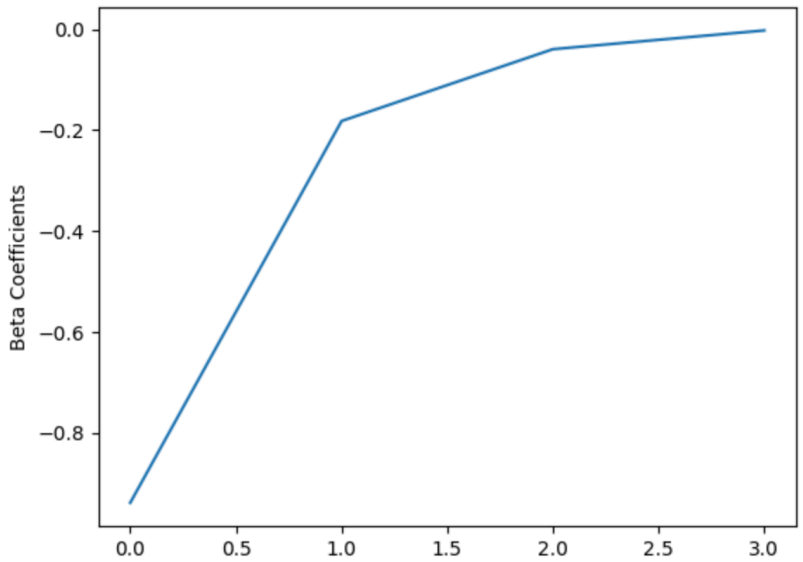
In [43]:
```
from pyspark.ml.classification import LogisticRegression

from pyspark.ml.classification import RandomForestClassifier
lr = RandomForestClassifier(featuresCol = 'features', labelCol = 'label')
# lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
lrModel = lr.fit(train)
trainingSummary = lrModel.summary
```

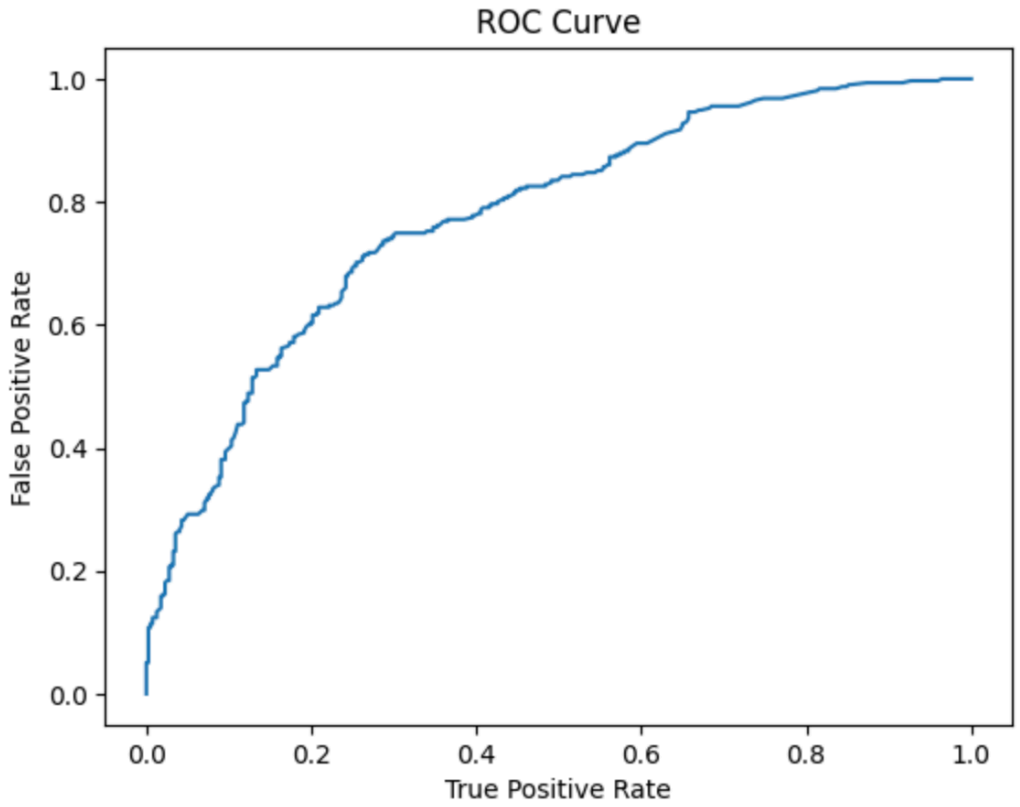
In [44]:
```
pr = trainingSummary.pr.toPandas()
plt.plot(pr['recall'],pr['precision'])
plt.ylabel('Precision')
plt.xlabel('Recall')
plt.show()
```




```
beta = np.sort(lrModel.coefficients)
plt.plot(beta)
plt.ylabel('Beta Coefficients')
plt.show()
```



```
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'],roc['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```




```
Training set areaUnderROC: 0.777321978329535
```