Laboratorio 4 Inteligencia de Negocios

Juan Alejandro Charry Gavilan – 201923748

1. Introducción:

En este laboratorio se busca implementar el modelo de regresión lineal del laboratorio 3 en un pipeline, de tal manera que sea consumible desde una API. Para lo anterior, se realizará la interfaz haciendo uso de FastAPI, pues es una manera rápida y sencilla para montar APIs.

Además de eso, para el pipeline se usará, como se mencionó anteriormente, el modelo del laboratorio 3. Sin embargo, se creará un pipeline para poder pasar los datos a través de un sistema de operaciones de orden lógico.

1. Datos

El uso y modelamiento de datos se mantuvo igual que en la entrega anterior, el único cambio que se realizó fue renombrar las variables del archivo Excel. Sin embargo, esto no cambia los resultados. Cabe resaltar que en el laboratorio 3 se observaron las variables lineales y su relación entre ellas. Por lo tanto, se puede decir que este laboratorio es una continuación del laboratorio 3.

1. Pipeline

En el pipeline estamos usando dos pasos; primero, ‘Imputer’ que toma los valores vacíos de las columnas y los remplaza con la media de los demás valores. Segundo, tenemos ‘std\_scaler’, que nos permite escalar los datos, ya que los algoritmos de machine learning de regresión lineal suelen funcionan mejor cuando se escalan los datos.

Además de esto, se encontraron usos adicionales de pipelines para datos cualitativos, no se necesitaron usar, ya que nuestro modelo sólo usa datos numéricos, sin embargo, se decidió mostrar el método y a continuación se explicarán los pasos. Primero, usamos ‘ordinal\_encoder’ para darle índices a cada valor único, segundo ‘OneHotEncoder’ que transforma los valores categóricos para que puedan ser manipulados por los usuarios.

El pipeline es el elemento crucial del laboratorio, es lo que permite realizar predicciones en el API de manera ligera, es en general, una serie de pasos que se realizan entre los datos, estos pasos optimizan el tiempo y espacio que se usa en el computador. Como se puede ver en los archivos .ipynb que se encuentran en la carpeta Pipeline, se mantuvieron los estudios de datos y se usaron para decidir finalmente cuales serían los hyperparametros seleccionados.

1. Modelo

Como es usual en los proyectos de machine learning, la parte más interesante se da en el modelo. Para este caso, se cambio el algoritmo de regresión que se estaba usando, en el laboratorio 3 se usó regresión lineal, la cual como se puede ver en los archivos. ipynb generaba un puntaje demasiado bajo, por lo que se optó por buscar nuevas alternativas.

Al investigar, se encontraron 5 algoritmos interesantes de regresión, por lo tanto se decidieron probar para ver cual era el mejor, estos algoritmos son: Decision tres/random forest, regresión robusta RANSAC, proceso de regresión gaussiano, regresión de vector de ayuda y regresión polinomial. Tras unas pruebas, que estuvieran sujetas al tiempo, se decidió que regresión polinomial tenía las estadísticas suficientes para cumplir con la tarea que se busca realizar. A continuación, se explicarán brevemente los algoritmos:

* 1. Radom forest:

Este algoritmo ya ha sido explicado de manera exhaustiva en anteriores laboratorios, por lo tanto, aquí se realizará una explicación general: random forest es un conjunto de arboles de decisión, cuando se desea clasificar un elemento, se ejecuta cada árbol del bosque y la categoría más repetida en los arboles es en la que se clasifica el objeto.

* 1. Regresión robusta:

Este método no es el mejor para el problema que se busca solucionar, sin embargo, es útil para modelos específicos, pues le asigna a los “outliers” (puntos muy lejanos al común; anormalidades) un gran peso. La razón por la que no se escogió es porque su fortaleza es su debilidad, en este caso no necesitamos tener en cuenta el comportamiento de los “outliers” para encontrar el comportamiento del común de hecho en modelos como este lo que se desea es eliminar las anormalidades para encontrar una media lo mas cercana a la realidad posible.

Este algoritmo sigue una idea similar a la de regresión lineal, pues se busca encontrar parámetros m y b que satisfagan:

Donde Y y X son conjuntos de puntos.

La principal diferencia entre regresión lineal y regresión robusta es que en regresión robusta se desea minimizar la formula:

Como se puede ver si se desea minimizar la suma de los cuadrados, un punto lejano dará una distancia mucho más grande. Se puede pensar como: al elevar al cuadrado un numero chico y uno grande, se separan mucho más.

* 1. Proceso de regresión gaussiano:

El proceso de regresión gaussiano se puede explicar con palabras de forma sencilla, se desea encontrar una función que pase por los puntos. Por lo tanto, tomamos todas las funciones y le asignamos una probabilidad a cada función. Como se puede ver el método gaussiano tiene mucha lógica cuando se explica así, el problema es expresarlo en términos matemáticos. Lo primero es que tenemos una distribución probabilística sobre cada función, esta distribución cambia de la siguiente manera:

La anterior ecuación sólo expresa que queremos encontrar la probabilidad de una función tal que llegan nuevos datos al sistema, esto será igual a la probabilidad de la función en todos los puntos, dividido en la probabilidad de observar esos puntos para una entrada específica.

Así que tenemos dos partes, la distribución de probabilidad de una función y tenemos una fórmula para adaptar la distribución a nuevos puntos entrantes. Podemos combinar las partes para construir un algoritmo, que calcula la distribución de funciones para datos de entrada.

Al final es muy probable que no encontremos una solución única, por lo tanto, podemos usar la media de todas las funciones. Sin embargo, ¿cómo podemos encontrar la media para funciones infinitas?, afortunadamente la función promedio es parte del proceso gaussiano y, por lo tanto, puede ser calculada. De hecho, los procesos de gauss son determinados por funciones de covarianza y la función de la media.

* 1. Regresión de vector de ayuda

Este método busca una línea recta que aproxima los valores medidos con una certeza dada. Si tenemos la función de una línea expresada cómo:

Si queremos encontrar la línea más plana entre los puntos podemos hacerlo minimizando la norma:

Podemos reescribir este problema como u ejercicio de optimización convexa:

* 1. Regresión polinomial:

Este fue el método elegido; el método es simple, de hecho la única diferencia con regresión lineal es el primero paso, ya que se genera una matriz que contiene todas las combinaciones de polinomios, para la entrada x.

Lo que hace es transformar el vector de entrada así:

Así, la función del modelo lineal queda:

1. Selección de modelo

Se seleccionaron tres modelos, esto porque nos pareció interesante usar el modelo de regresión polinomial, usando un ciclo de for encontramos el óptimo, que resultó ser con un polinomio de grado 12. Para ver esto se puede entrar al archivo de “Estudio datos para generar Pipeline” en la carpeta de Pipeline. Es importante que al revisar los modelos se compararon usando tres métricas. Sin embargo, debido a que no había competencia con el modelo óptimo, pues tenía los mejores valores en cada métrica, no se necesitó elegir una métrica para comparar modelos.

* 1. Explained Variance Score

Se buscan valores cercanos a 1 son deseados, ya que si restamos y con y\_{estimado} y son parecidos (que es lo que se espera) el numerador dará cerca a cero, lo que generara que la función sea cercana a uno, el divisor esta solo para volver el valor porcentual.

* 1. Max\_error

Como dice el nombre, esta funcion retorna el error maximo recidual entre y\_{estimado} y y. Es importante resaltar que este dato puede ser afectado por una anormalidad, por lo que usualmente no es una buena métrica para medir algoritmos.

* 1. mean\_absolute\_error (MAE)

Basicamente nos da el primedio de errores cometidos, como podemos ver el resultado es 8.3 debido a que el modelo está en una escala donde el mayor es 101.96 y el menor es 31.55 (como se puede observar abajo), por lo tanto un error promedio de 8 puntos parece no ser demasiado grande.

Cabe reconocer que existen muchas más métricas interesantes, se recomienda mirar la documentación en el siguiente enlace: [API Reference — scikit-learn 1.1.3 documentation](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics). Ya que no sólo están documentados lo métodos, sino que también están agrupados por la categoría de cada algoritmo.

Por último, se quiere recalcar que el valor cuantitativo de las métricas está en archivo de “Estudio datos para generar Pipeline” por cada modelo analizado.

1. API

No se tuvieron muchos problemas para montar el API, el framework de FastAPI nos pareció excelente, a continuación, se muestran pantallazos de las pruebas de postman por cada método.

* 1. predictModel1:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. predictModel2:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. predictModel3:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. predictionsModel1

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* 1. predictionsModel2

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. predictionsModel3

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. scoreModel1

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. scoreModel2

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated with medium confidence

* 1. scoreModel3

Graphical user interface, text, application, email, website

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. trainModel1

Graphical user interface, text, application

Description automatically generated

* 1. trainModel2

Graphical user interface, text, application

Description automatically generated

* 1. trainModel3

Graphical user interface, text, application, Word

Description automatically generated

Se desea recalcar que cada vez que se reentrena un modelo o se pide el puntaje de un modelo, lo que se hace es retornar el valor de predicción (Explained Variance Score) y valor absoluto medio (MAE).

1. Conclusiones

El problema presentado aún tiene mucho campo para explorar, sin embargo, se piensa que se llegó a una buena solución debido a que los puntajes recibidos por los modelos son muy buenos, la mayoría están sobre 0,9. Si el negocio cuenta con más tiempo, se recomienda seguir explorando posibilidades.

Por otro lado, el mejor modelo que se obtuvo tenía un puntaje de 0,99, lo cual es muy inusual y de hecho genera dudas sobre el modelo en sí, ya que cuando un modelo tiene valores tan altos, se suele pensar que se cometió un error. Pero, se realizó una revisión exhaustiva de cada método y lo que se estaba haciendo, por ende, se llegó a la conclusión que se encontró un algoritmo que destaca mucho con estos datos.