New York City Bus

Paula Oviedo ~ Nicolás Patalagua

paulaj.oviedo@correo.usa.edu.co ~ alvaro.patalagua@correo.usa.edu.co

https://github.com/NicoPatalagua/BigData/blob/master/mta_1712.ipynb

https://github.com/NicoPatalagua/BigData/blob/master/proyecto2.ipynb

Resúmen: *En este documento se realizará un análisis con técnicas de big data y aplicación de ciertas herramientas de inteligencia artificial a un dataset publicado en la comunidad científica Kaggle,*

Abstract: ***T****his document will carry out an analysis with big data techniques and the application of certain artificial intelligence tools to a dataset published in the Kaggle scientific community.*

I. MARCO TEÓRICO

MTA Bus Company es conocido a nivel mundial por ser el servicio de la MTA Regional Bus Operations, usado en las rutas previamente operadas por el New York City Department of Transportatión, responsable del manejo de la infraestructura de transporte de la ciudad de Nueva York. [1]
Actualmente el NYCDOT o departamento de transporte de la ciudad de Nueva York, tiene su sede en el 55 water street (Nueva York), el alcalde de la ciudad en el periodo 2002 a 2013, eligió como comisionado a Janette Sadik-Khan, elegido el 27 de abril de 2007.[2]

Modelo 3D de bus del MTA Bus Company

El departamento de transporte, no solo incluye transporte sino a su vez el mantenimiento continuo de la infraestructura de movilidad de la ciudad, a parte de esto realiza la instalación y mantenimiento de letreros, señales de tránsito, semáforos, letreros de direcciones, instalación de parquímetros, tendido eléctrico y en su gran mayoría la red de estacionamiento público. Entre otras de sus tareas está el ofrecer el servicio fluvial de transbordador entre la isla de Manhattan y el distrito de staten Island, este servicio es comúnmente conocido como el Ferry de Staten Island.[3]

Por otra parte el MTA New York City Bus, tema principal del dataset es un servicio de autobuses que opera en las cinco divisiones administrativas de la ciudad que a nivel de condado son llamadas *boroughs*, estas son *Bronx, Brooklyn, Manhattan, Queens y Staten Island.* A día de hoy cuenta con una flota de 4500 autobuses y 200 rutas locales, sirviendo de complemento a las líneas de ferrocarril de la *MTA*, al metro de Nueva York, el ferrocarril de Staten Island, el ferrocarril de Long Island y el Ferrocarril de Metro-North.[4]
Con el fin de mejorar las actualizaciones de tráfico en vivo, la *MTA Bus Company,* creó el *MTA Bus Time,* la mayoría de datos contenidos en el dataset ***New York City Bus Data***[5]. *MTA Bus Time,* es una aplicación con interfaz de servicio que brinda información en tiempo real, un sistema de información de pasajeros y una ubicación automática del vehículo comúnmente conocida como AVL, esta aplicación desarrollada por la *Autoridad Metropolitana de Transporte (**MTA**)* . Esta aplicación es comúnmente usada por ciudadanos y turistas en las operaciones de autobuses bajo el *Nuevo York City Bus* y *MTA Bus Company*. Fue lanzada en enero de 2011 y se implementó en su totalidad en 2014, usando tecnologías de posicionamiento global como GPS.[6]
Ahora bien el dataset es el resultado de un flujo de datos opensource que fue elegido luego de buscar entre otros dataset, dado que a nivel mundial muchos buses transmiten su ubicación GPS y otros datos en vivo, en busca de una fuente ideal de datos se escogió el dataset de la *MTA* en Nueva York al ser una empresa confiable y con los datos que se necesitan.
El contenido de datos de esta serie de 4 dataset en formato de *CSV*, o archivo separado por comillas. Los conjuntos de datos se obtienen del servicio de flujo de datos de los autobuses NYC MTA, a través de la aplicación ya mencionada.[5]
Contienen crecimientos de aproximadamente 10 minutos, la ubicación, rutas, paradas del autobús y otros datos en cada fila. También se incluye la hora de llegada programada del horario del autobús, con el fin de dar una indicación de dónde estará el autobús, el tiempo de retraso e incluso si llega antes de lo previsto. Aparte de la aplicación de clientes, los datos también fueron recolectados con la aplicación *MTA SIRI* y los datos del programa *MTA GTFS*.[5]
Entre otros datos acerca del dataset, cuenta con 25700 vistas y 2687 descargas, lo que significa un promedio de 0.1 descargas por vista, con un total de 6 colaboradores y un pico de actividad entre mayo y junio del 2019. Cuenta con 5 kernels de los cuales solo uno tiene comentarios sin embargo ninguno tiene medalla de clasificación en *Kaggle*, la usabilidad es de cerca de 7.1 porque cuenta con subtítulos, etiquetas, descripción, imagen de portada, es público, pero no cuenta con descripción de columnas ni una frecuencia de muestreo específico.[5]

Entre las columnas definidas en el conjunto de datos se encuentran las siguientes:

a. Recorded at Time
b. Direction Ref
c. Published Line Name
d. Origin Name
e. Origin Lat
f. Origin Long
g. Destination Name
h. Destination Lat
i. Destination Long
j. Vehicle Ref
k. Vehicle Location Latitude
l. Vehicle Location Longitude
m. Next Stop Point Name
n. Arrival Proximity Text
o. Distance From Stop
p. Expected Arrival Time
q. Scheduled Arrival Time

Como se había mencionado existen 4 dataset, el primero tiene un tamaño de 1.32 Gb, el segundo 1.26 Gb, el tercero 1.34 Gb y finalmente un cuarto dataset con 1.24 Gb. Se unieron estos dataset respectivamente para realizar la obtención de nuevos elementos.[5]

II. BIG DATA

Después de una introspección de cada columna, se tomó la decisión de depurar el Dataset. Para esto se eliminaron columnas que no aportan valor como lo son las siguiente:

- Origin Lat
- Origin Long
- Destination Lat
- Destination Long
- Vehicle Location Latitude
- Vehicle Location Longitude

Luego de la limpieza del DataSet se procedió a realizar las consultas que fueron las siguientes:

- Las estaciones más concurridas
- Conteo de los trayectos realizados por dia
- Los vehículos con más trayectos realizados
- Vehículos con más trayectos por cada mes
- Tiempo promedio de se tarda entre estaciones

III. INTELIGENCIA ARTIFICIAL

RapidMiner conocido anteriormente como *Yet Another Learning Environment,* es un software opensource de análisis y minería de datos, en este programa se permite realizar el análisis de datos a través del encadenamiento de operadores con un entorno gráfico. [7]

Para efectos de este trabajo se hizo uso de rapidMiner con el fin de realizar el diseño de modelos predictivos, debido a que este es uno de los primeros frentes del trading cuantitativo. Tal como lo dice Tradingsys.org, las plataformas basadas en redes neuronales e incluso modernos generadores automatizados de código han realizado esfuerzos por conseguir que los datos realicen predicciones por sí solos. Es aquí donde se evidencia *data mining*, la minería de datos son un conjunto de técnicas estadísticas de información que permite extraer de manera automática patrones ocultos, tendencias, vinculaciones, entre otras características en grandes volúmenes de datos.

En la minería de datos se pueden aplicar algoritmos de regresión, de clasificación, de segmentación, asociación, interpretación e incluso de series temporales. Para el análisis que se realizará en este documentos se usarán algoritmos de clasificación, con el fin de realizar predicciones sobre variables discretas relacionadas con el dataset *MTA*, algunos son algoritmos de agrupamiento que evidencien resultados por distancias media entre los elementos del grupo u otras características relevantes.

Entre las opciones de algoritmos que se le aplicarán al dataset, se encuentra *AdaBoost,, Bagging, Extra Trees, Gradient Boosting, Isolation Forest, RandomForest,* entre otros, la mayoría de ellos con la aplicación de árboles de decisión.

Cross Validation: Esta técnica consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones.[8]

Decision Tree: Dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema.[9]

Naive Bayes (Gaussian): un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales. Es a causa de estas simplificaciones, que se suelen resumir en la hipótesis de independencia entre las variables predictoras, que recibe el apelativo de *ingenuo*.[10]

Random Forest: Los bosques aleatorios son una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia.[11]

Support Vector Machine: Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación definido como el vector

entre los 2 puntos, de las 2 clases, más cercanos al que se llama vector soporte. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase.[12]

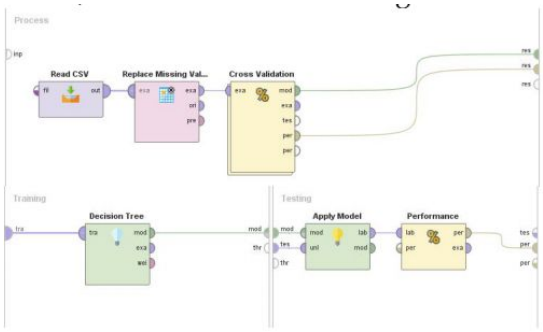
Adaboost: Consiste en combinar los resultados de varios clasificadores débiles para obtener un clasificador robusto. un clasificador robusto es un clasificador que tiene un mejor desempeño que el de un clasificador débil, ya que sus clasificaciones se aproximan más a las verdaderas clases.[13]

Bagging: Agregación de bootstrap, es un meta algoritmo de aprendizaje automático diseñado para mejorar la estabilidad y precisión de algoritmos de aprendizaje automático usados en clasificación estadística y regresión. Además reduce la varianza y ayuda a evitar el sobreajuste. Aunque es usualmente aplicado a métodos de árboles de decisión, puede ser usado con cualquier tipo de método. El empaquetado es un caso especial del promediado de modelos.[14]

Perceptron: es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple). El perceptrón multicapa puede estar totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa "i" es entrada de todas las neuronas de la capa "i+1", mientras que en el segundo cada neurona de la capa "i" es entrada de una serie de neuronas (región) de la capa "i+1".[15]

Rappidminer:

1. **Decision Tree:** La construcción del árbol de decisión, se realiza de la siguiente forma:

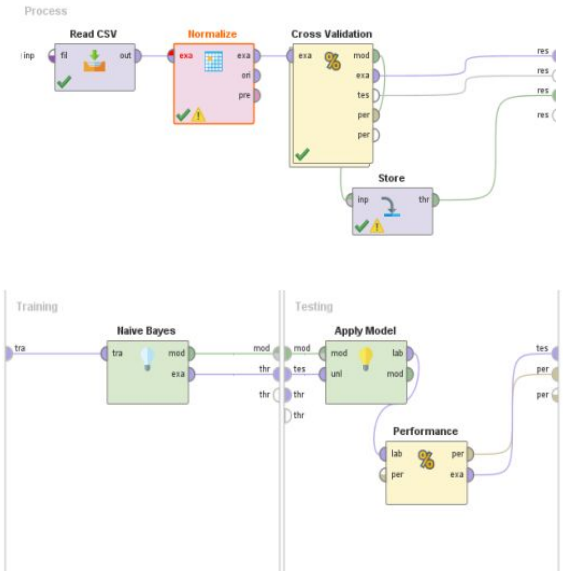


Decision Tree con Cross validation

Al crear un árbol de decisión con una profundidad de 7 ramas, el modelo aumenta complejidad provocando un overfitting.Entre más profundo sea el árbol, se puede entrenar partiendo de diversas carácterısticas, llegando a un atributo del 100%.

La Validación Cruzada o k-fold Cross Validation consiste en tomar los datos originales y crear a partir de ellos dos conjuntos separados: un primer conjunto de entrenamiento (y prueba), y un segundo conjunto de validación. El modelo se entrena en el conjunto de entrenamiento y se puntúa en el conjunto de prueba. Luego, el proceso se repite hasta que cada grupo único se haya utilizado como conjunto de prueba. El operador de validación dividida es un operador anidado. Tiene dos subprocesos: un subproceso de capacitación y un subproceso de prueba. El subproceso de capacitación se utiliza para aprender o construir un modelo.

2. **Naive Bayes:**
En el Pipelinde rapidminer del dataset de lens24 age=3, specRx=1, astig=2, tears = 2: Se usó el bloque store en cual consiste:Este operador almacena un objeto IO en una ubicación en el repositorio de datos. La ubicación del objeto a almacenar se especifica a través del parámetro de entrada del repositorio. El objeto almacenado puede ser utilizado por otros procesos mediante el uso del operador Recuperar. Después de entrenar el modelo su estructura de la siguiente forma para los distinto modelos.



Naive Bayes con cross validation

En rapidminer se usó un dataset de 1.26 Gb, y usando las columnas Recorded at Time como label principal, Direction Ref, Published Line Name, Origin Name, Destination Name,Vehicle Ref, Next Stop Point Name, Arrival Proximity Text, Distance From Stop y Expected Arrival Time. Se realizó un análisis con *support vector machine, decision tree, Naive bayes, Adaboost, Random forest y bagging.* Las predicciones siguientes se hicieron entre *Recorded at Time* y *Expected Arrival Time.*

RecordedAtTime	ExpectedArrivalTime
2017-12-01 00:05:49	2017-12-01 00:06:21
2017-12-01 00:06:19	2017-12-01 00:06:21
2017-12-01 00:05:25	2017-12-01 00:06:22
2017-12-01 00:05:56	2017-12-01 00:06:22
2017-12-01 00:06:02	2017-12-01 00:06:24
2017-12-01 00:06:15	2017-12-01 00:07:54
2017-12-01 00:06:07	2017-12-01 00:06:31
2017-12-01 00:05:50	2017-12-01 00:06:39

Columnas usadas para predicciones

accuracy: 97.84% +/- 0.24% (micro average: 97.84%)

	true 0	true 1	class precision
pred. 0	16180	308	98.13%
pred. 1	79	1331	94.40%
class recall	99.51%	81.21%	

Support Vector Machine

accuracy: 97.70% +/- 0.39% (micro average: 97.70%)

	true 0	true 1	class precision
pred. 0	16117	269	98.36%
pred. 1	142	1370	90.61%
class recall	99.13%	83.59%	

Decision Tree

accuracy: 94.50% +/- 0.77% (micro average: 94.50%)

	true 0	true 1	class precision
pred. 0	15525	250	98.42%
pred. 1	734	1389	65.43%
class recall	95.49%	84.75%	

Naive Bayes

accuracy: 97.87% +/- 0.26% (micro average: 97.87%)

	true 0	true 1	class precision
pred. 0	16171	294	98.21%
pred. 1	88	1345	93.86%
class recall	99.46%	82.06%	

Adaboost

accuracy: 97.93% +/- 0.32% (micro average: 97.93%)

	true 0	true 1	class precision
pred. 0	16179	291	98.23%
pred. 1	80	1348	94.40%
class recall	99.51%	82.25%	

Random Forest

accuracy: 97.85% +/- 0.25% (micro average: 97.85%)

	true 0	true 1	class precision
pred. 0	16176	302	98.17%
pred. 1	83	1337	94.15%
class recall	99.49%	81.57%	

Bagging

Al realizar una comparación del accuracy, recall y precision que genera cada uno de los modelos, encontre que Support Vector Machine y Random Forest son similares y eficientes, sin embargo el mejor algoritmo fue Random forest con un accuracy de 97.93%, un recall de 99.51% y una precisión de 98.23%. Luego de realizar estos análisis vemos que en términos de accuracy el modelo de random forest, genera mejores resultados a razón de que tiene una capacidad de generalización muy alta para muchos problemas.

Sklearn:

Luego de haber hecho un análisis previo en rapidminer, vamos a usar scikit-learn con varios kernel de modelos de clasificación y una comparación entre ellos, los datos de comparación se deben al puntaje del accuracy, al puntaje macro y micro de la precisión, al puntaje macro y micro del recall y por último al reporte macro y micro del nF1 de cada uno de los modelos.

Los resultados obtenidos fueron lo siguientes:

1. Decision Tree:

```
Decision Tree Classifier
Accuracy Score: 0.03303237172428981
Precision Score Macro: 0.21328781634828187
Precision Score Micro: 0.03303237172428981
Recall Macro: 0.13907986044645482
Recall Micro: 0.03303237172428981
F1 Score Macro: 0.02140826327937601
F1 Score Micro: 0.03303237172428981
```

2. Random Forest Classifier

```
Random Forest Classifier
Accuracy Score: 0.024003523452983924
Precision Score Macro: 0.2110548707012439
Precision Score Micro: 0.024003523452983924
Recall Macro: 0.12520853424032616
Recall Micro: 0.024003523452983924
F1 Score Macro: 0.012979013842919935
F1 Score Micro: 0.024003523452983924
```

3. Random Forest Classifier Mejorado:

```
New Random Forest Classifier
Accuracy Score: 0.020700286280554942
Precision Score Macro: 0.2125768072724148
Precision Score Micro: 0.020700286280554942
Recall Macro: 0.1169050220360264
Recall Micro: 0.020700286280554942
F1 Score Macro: 0.011488387023298987
F1 Score Micro: 0.020700286280554942
```

4. Gaussian NB Classifier

```
GaussianNB Classifier
Accuracy Score: 0.020700286280554942
Precision Score Macro: 0.5411412788983038
Precision Score Micro: 0.020700286280554942
Recall Macro: 0.07057552914695772
Recall Micro: 0.020700286280554942
F1 Score Macro: 0.010610531967110976
F1 Score Micro: 0.020700286280554942
```

5. Bagging Classifier

```
Bagging Classifier
Accuracy Score: 0.03171107685531821
Precision Score Macro: 0.2500949241290027
Precision Score Micro: 0.03171107685531821
Recall Macro: 0.11831236209464993
Recall Micro: 0.03171107685531821
F1 Score Macro: 0.02210105068886367
F1 Score Micro: 0.03171107685531821
```

6. Bagging Classifier Mejorado

```
New Bagging Classifier
Accuracy Score: 0.029068487117375027
Precision Score Macro: 0.223706026919386 65
Precision Score Micro: 0.029068487117375027
Recall Macro: 0.15440557283313466
Recall Micro: 0.029068487117375027
F1 Score Macro: 0.019624917013609418
F1 Score Micro: 0.029068487117375027
```

7. AdaBoost Classifier

```
AdaBoost Classifier with Decision Tree Clssifier
Accuracy Score: 0.02048007046905968
Precision Score Macro: 0.2143654675916575
Precision Score Micro: 0.02048007046905968
Recall Macro: 0.12565867585651758
Recall Micro: 0.02048007046905968
F1 Score Macro: 0.012048543801364994
F1 Score Micro: 0.02048007046905968
```

8. Perceptron

```
Percptron Classifier
Accuracy Score: 0.002422373926447919
Precision Score Macro: 0.9628871144551706
Precision Score Micro: 0.002422373926447919
Recall Macro: 0.007591969783204843
Recall Micro: 0.002422373926447919
F1 Score Macro: 0.0003132693230498308
F1 Score Micro: 0.002422373926447919
```

9. Perceptron Mejorado

```
Accuracy Score: 0.002422373926447919
Precision Score Macro: 0.9628871144551706
Precision Score Micro: 0.002422373926447919
Recall Macro: 0.007591969783204843
Recall Micro: 0.002422373926447919
F1 Score Macro: 0.0003132693230498308
F1 Score Micro: 0.002422373926447919
```

Ahora bien los datos proporcionados por estos modelos incluyen la precisión, el recall o exhaustividad, el valor F o F1 score, la exactitud o accuracy, se podría graficar la matriz de confusión con los positivos verdaderos y falsos y los negativos positivos y falsos.

Nos centraremos en los siguientes:

a. Precisión: Con la cual se puede medir la calidad del modelo de clasificación, en este caso la precisión responde a la pregunta ¿Qué porcentaje de buses esperados llegan en el tiempo indicado en con el horario de llegada?
b. Recall: LA exhaustividad nos mide la cantidad que el modelo es capaz de identificar, en este caso ¿Cuántos buses esperados que llegan a tiempo en base al tiempo de llegada estimada, somos capaces de identificar?
c. F1-score: Es una combinación de la precision y recall en un solo valor. Con el fin de comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.
d. Accuracy: la exactitud del porcentaje de casos en los cuales el modelo es acertado, sin embargo puede llevarnos al engaño haciendo un modelo como malo o que un modelo malo parezca mejor a lo que es.

IV. REFERENCIAS

[1]fliwanag, «Staten Island/MTA City Transit Bus». https://3dwarehouse.sketchup.com/model/b70e1062c67b159367f73e2a1cdb19c9/Staten-IslandMTA-City-Transit-Bus (accedido abr. 29, 2020).

[2]«New York City Bus Schedules», *MTA*. https://new.mta.info/schedules/bus (accedido abr. 29, 2020).

[3]«Facts and Figures», feb. 18, 2010. https://web.archive.org/web/20100218032806/http://www.mta.info/nyct/facts/ffbus.htm (accedido abr. 29, 2020).

[4]«MTA/New York City Transit Bus Schedules», ene. 20, 2010. https://web.archive.org/web/20100120062308/http://www.mta.info/nyct/service/bus/bussch.htm (accedido abr. 29, 2020).

[5]«New York City Bus Data». https://kaggle.com/stoney71/new-york-city-transport-statistics (accedido abr. 29, 2020).

[6]«MTA Bus Time». https://bustime.mta.info/ (accedido abr. 29, 2020).

[7]«RapidMiner | Best Data Science & Machine Learning Platform», *RapidMiner*. https://rapidminer.com/ (accedido abr. 29, 2020).

[8]«3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/cross_validation.html (accedido abr. 29, 2020).

[9]«1.10. Decision Trees — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/tree.html (accedido abr. 29, 2020).

[10]«1.9. Naive Bayes — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/naive_bayes.html (accedido abr. 29, 2020).

[11]«1.11. Ensemble methods — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees (accedido abr. 29, 2020).

[12]«1.4. Support Vector Machines — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/svm.html (accedido abr. 29, 2020).

[13]«sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html?highlight=adaboost#sklearn.ensemble.AdaBoostClassifier (accedido abr. 29, 2020).

[14]«sklearn.ensemble.BaggingClassifier — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html?highlight=bagging#sklearn.ensemble.BaggingClassifier (accedido abr. 29, 2020).

[15]«sklearn.linear_model.Perceptron — scikit-learn 0.22.2 documentation». https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html?highlight=perceptron#sklearn.linear_model.Perceptron (accedido abr. 29, 2020).