



Equipo 9

Predicción de Salarios para áreas de Analítica de Datos, ML e IA

Maestría en Inteligencia Analítica de Datos – MIAD
Despliegue de soluciones Analíticas

Alejandra Barbosa
Paulina Luissi
Juan Camilo Quintana
Johanna Katherine Sepulveda Villamizar

Bogotá D.C, Colombia
2024-11-26

1. Resumen del problema.

Actualmente el mercado laboral para posiciones relacionadas con Análisis de Datos (AD), Machine Learning (ML) e Inteligencia Artificial (IA) se está volviendo cada vez más competitivo debido a la importancia que juegan estos roles en las compañías. Por lo tanto entender y predecir adecuadamente las tendencias salariales para estas posiciones es esencial tanto para las compañías que quieren atraer y retener el talento, como para aquellos que estén buscando oportunidades de empleo en estas áreas puedan definir adecuadamente sus expectativas salariales al momento de tomar decisiones entre las distintas oportunidades laborales. Esto es cada día más relevante a medida que este mercado está cada vez más activo, y la modalidad de trabajo virtual permite que personas ubicadas en distintos puntos geográficos trabajen de manera remota.

Preguntas de negocio

¿Qué salario puedo esperar obtener en las posiciones de AD, ML e IA y según las características de la posición disponibles a elección del usuario?

¿Qué salario deberíamos ofrecer como empresa en las posiciones de AD, ML e IA según las características de la posición disponibles con el objetivo de plantear un salario competitivo y atractivo para los postulantes?

¿Cómo varían los salarios para estas posiciones en las distintas regiones y países?

Alcance del proyecto

El proyecto comprende la implementación de modelos de machine learning supervisados y su posterior evaluación y selección para incorporar en una aplicación. La aplicación se trata de un tablero de control a través del cual tanto usuarios individuales como empresas puedan predecir e identificar tendencias salariales en los roles objeto de este proyecto.

Para la implementación de los modelos y de la aplicación está se incorporarán prácticas de MLops y de despliegue de aplicaciones cómo son: versionamiento de datos y del código, integración mediante APIs y plataformas abordadas en el curso que se irán incorporando a medida que avance el ciclo y logremos un entendimiento que nos permita elegir las herramientas a utilizar.

Descripción de conjuntos de datos a emplear.

Los datos consisten en información salarial de roles relacionados con AI y Machine Learning, extraída de la pagina <https://aijobs.net/salaries/download/>, estos son de dominio público, y estan licenciados por CC0 1.0 (Universal Deed) lo cual significa que **aijobs.net**, la compañía que los administra, ha renunciado a sus derechos de autor, de modo que éstos pueden ser copiados, modificados, distribuidos y utilizados aún con fines comerciales sin necesidad de pedir permiso. Asimismo, estos datos no tienen garantía alguna, y **aijobs.net** no tiene ninguna responsabilidad o aprobación sobre el uso que den terceros a éstos. A continuación se presentan las variables que conforman los datos:

- Año en el cual se recibe el salario, Nivel de experiencia (Junior, Intermedio, Experto, Director/Ejecutivo), Tipo de Empleo (Medio tiempo, Tiempo completo, Contrato, Freelance), Título de la Posición (varias opciones), Salario (moneda local), Moneda salario, Salario en USD (calculado por **aijobs.net**), Residencia del empleado, Tasa de trabajo remoto (Presencial: Menos 20%, Híbrido: Entre el 20% y 80%, Remoto: Más del 80%), Ubicación de la Compañía (Oficinas Principal, o sucursal que contrata), Tamaño de la Compañía (Pequeña: Menos de 50 empleados, Mediana: 50 - 250 empleados, Grande: Más de 250 empleados)

Selección de Características

La selección preliminar de características fue realizada en la exploración de datos. En esta se determinó que las variables a utilizar serían todas categóricas (incluida `remote_ratio` pues son valores 0, 50 o 100 que pueden ser vistos como categóricas). Estas variables finales fueron además de la variable de respuesta `salary_in_usd`:

- `experience_level`
- `employment_type`
- `job_title`
- `employee_residence`
- `company_location`
- `company_size`

Durante la construcción de los modelos de clasificación se tuvo en cuenta también la importancia de las variables para depurar los modelos.

Manejo de versionamiento de datos

Para el manejo de datos, los diferentes archivos csv fueron versionados con DVC. Con dvc se crea un “hash” del contenido del archivo .csv. Tener archivos en DVC en lugar de rastrearlos directamente con Git es útil porque DVC está diseñado específicamente para manejar grandes archivos de datos y modelos, algo que Git no hace de manera eficiente.

Para gestionar los datos del proyecto con DVC, se utilizó una instancia de Ubuntu en AWS. Esta elección permite trabajar en un entorno controlado, escalable y con recursos ajustables según las necesidades del proyecto. Se lanzó una instancia EC2 en AWS con el sistema operativo Ubuntu, seleccionando un tipo de instancia acorde al tamaño y complejidad de los datos (en este caso t2.micro).

En el repositorio, se puede ver que se tienen tanto los datos versionados .dvc dentro de la carpeta Data/data/, y los mismos archivos en .csv en Data/. La razón de esto es que, al no ser datos pesados y falta de conocimiento para leer datos directamente desde dvc para incluirlo en los modelos, se decidió guardar ambos tipos de archivo. De esta manera se garantiza la lectura de los datos fácilmente con raw github contents al momento de crear el dashboard, pero de igual manera se tiene un control de versionamiento de datos con DVC.

La separación entre los archivos .csv y los .dvc permite tener un acceso claro a los datos crudos para visualización rápida y prototipado, mientras que el uso de DVC permite reproducir resultados y seguir la trazabilidad de cambios en experimentos.

2. Modelos desarrollados y su evaluación.

Este problema de predicción se puede abordar como uno de regresión o de clasificación. Decidimos dividir la implementación de los modelos así:

Tipo de modelo	Modelo	Variable Objetivo
Regresión	XGBoost	salary_in_usd (continua)
	LGBM	
	CatBoost	
Clasificación	LGBM CatBoost Random Forest	1. salary_in_usd (expresada en 4 rangos): Bajo < 75000 Medio Bajo (75000 - 120000) Medio Alto (120000 - 180000) Alto > 180.000
	XGBClassifier	2. salary_in_usd expresada en 3 Rangos, nombrado Terciles < 117000 117000 - 177000 > 177000

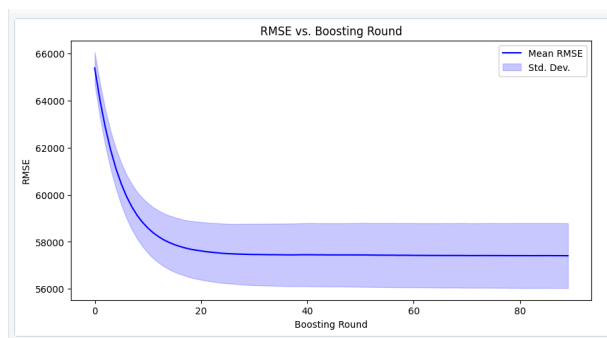
Para la preparación se usó el mismo set de datos y se separó la base en entrenamiento y prueba, con el fin de evaluar el desempeño de los modelos. Para regresión se utilizó la variable salario continua y para clasificación se separó en diferentes clases.

modelo devuelve el salario posible de una persona que desea trabajar en ciertos salarios con una diferencia de más o menos 56.020 USD. Algo que podríamos decir no es muy bueno modelo y falta ser refinado.

Light Gradient Boosting Machine

Este modelo también está basado en árboles de decisión como XGBoost pero se diferencia de éste en que sigue una estrategia basada en hojas. En vez de buscar todas las posibles divisiones para cada árbol se expande sobre las hojas que generan una mejor ganancia para el modelo. Este tiene una capacidad de captar relaciones no lineales y cuenta con la ventaja de poder trabajar con variables categóricas sin tener que transformarlas, para ello es necesario indicar que son categóricas antes de correr el modelo. En este caso lo implementamos en un contexto de regresión con el objetivo de predecir el salario anual correspondiente a ciertas variables explicativas. El set de datos utilizado para correr el modelo ya fue limpiado previamente en la etapa de exploración de datos y ya cuenta con las variables que definimos utilizar para trabajar los modelos: `experience_level`, `employment_type`, `job_title`, `employee_residence`, `company_location`, `company_size` y `remote_ratio`.

Para la evaluación del modelo se utiliza el RMSE al igual que con XGBoost.



Al inicio los parámetros seleccionados son aleatorios pero inmediatamente procedemos a correr un Cross validation para definir un número óptimo de iteraciones (rounds) que queremos deberíamos correr para evitar el sobreajuste, esto da como resultado 20 rounds

Con los 20 rounds y unos parámetros iniciales el RMSE obtenido es 55815 USD

Considerando que el promedio de salario es 153546 USD consideramos que el resultado tiene margen de mejora, siendo que el error es del 36%.

Para mejorar el desempeño implementamos Grid Search con Cross validation de 5 folds para poder determinar los mejores parámetros para correr el modelo (notebook en github `ModeloLGBM_Completo.ipynb`)

Se prueba en Databricks el modelo con distintos hiper parámetros siendo el de mejor performance “unique-goat-227”

Experiments > ModeloLGBM_9_11_2024											
Add Description											
Q metrics.rmse < 1 and params.model = "tree"											
Time created State: Active Datasets Sort: Created Columns Expand rows Group by											
Table	Chart	Evaluation	Preview								
	Run Name	Created	Dataset	Duration	Source	Models	Metrics		Parameters		
							best_iteration	valid_0_rmse	early_stopping	learning_rate	max_depth
	placid-snail-225	13 days ago	dataset (8c9c758)	6.5s	ModeloL...	lightgbm	40	55633.92526...	10	0.1	6
	unique-goat-227	13 days ago	dataset (8c9c758)	6.0s	ModeloL...	lightgbm	20	55643.24478...	10	0.3	4
	unruly-goose-552	13 days ago	dataset (8c9c758)	6.2s	ModeloL...	lightgbm	19	55702.62986...	10	0.25	4
	unleashed-squid-662	13 days ago	dataset (8c9c758)	7.6s	ModeloL...	lightgbm	20	55815.28164...	10	0.1	6

num_leaves = 5

Este hiper parámetro se refiere al número de nodos que cada árbol puede tener y esto define qué grado de complejidad puede capturar el modelo.

learning_rate = 0.3

Con este definimos qué tanto va a contribuir cada árbol al modelo total y este está estrechamente relacionado con los próximos parámetros. En este caso es un learning rate alto considerando que el número de iteraciones es bajo.

max_depth = 4

Este valor se refiere a la profundidad que puede alcanzar el árbol y en este caso se trata de un valor bajo.

min_data_in_leaf = 20

Este valor define cuantos datos deben ser representados en cada hoja final y el valor bajo está destinado a evitar el sobreajuste.

num_boost_round = 20

Este valor se refiere a la cantidad de iteraciones que realizará el modelo. Es también un valor bajo.

Con estos parámetros el RMSE baja mínimamente a 55643 USD y se mantiene en el 36% del salario promedio. Estos parámetros seleccionados con Grid Search parecen ser adecuados para un modelo de baja complejidad lo que resulta razonable siendo que se cuenta con muy pocas variables e incluso en los modelos de clasificación se pudo observar que algunas de ellas no aportan un valor significativo.

Al intentar correr una versión que permita captar mayor complejidad “placid-snail-225” (menor learning rate, mayores iteraciones, profundidad y número de hojas) notamos que el desempeño no mejora.

CatBoostRegressor (Modificado en esta entrega)

CatBoost es un algoritmo de Machine Learning basado en gradient boosting, que se distingue de otros modelos de Árboles de Decisión basados en gradient boosting (GBDTS: Gradient Boosted Decision Trees) al ofrecer soluciones únicas para interpretar fuentes de datos que son fundamentalmente categóricas o contienen muchos datos faltantes[1], algunas de las ventajas que presenta este modelo son que no es necesario el preprocesamiento de las variables categóricas, está diseñado para ser rápido y eficiente, en donde ha probado entrenar modelos más rápido que un modelo de XGBoost y LightGBM y finalmente es un modelo robusto y preciso [2]. Los parámetros utilizados fueron los siguientes [3]:

- **Depth:** Profundidad máxima de los árboles de decisión que se crean, es decir controla que tan complejo es el árbol de decisión, en donde a mayor profundidad se puede capturar mejor la complejidad de las características de los datos sin embargo también hay que tener en cuenta que se puede generar un sobreajuste al modelo
- **Iterations:** Máximo número de árboles que se crean para entrenar el modelo. Define que tanto aprende el modelo, ya que con un valor muy bajo el modelo podría generar que no aprenda bien las complejidades de los datos, pero un valor muy alto podría generar sobreajuste
- **Learning Rate:** Es la tasa de aprendizaje, esto controla el tamaño del paso del gradiente para minimizar la función de costo, en la medida que este se hace menor el algoritmo se tardará más en hacer el entrenamiento. Además si es demasiado pequeña se puede dar overfitting porque aprende demasiado y si es demasiado grande puede que el algoritmo no converja.
- **L2 Leaf Reg:** Es un parámetro de regularización, que se usa con el fin de reducir el sobreajuste mediante la penalización de los coeficientes de los árboles en el entrenamiento donde un valor pequeño permite que el modelo se ajuste sin muchas “restricciones” a los datos de entrenamiento lo que a su vez puede generar sobreajuste, pero un valor muy grande puede causar que el modelo no se ajuste adecuadamente a la complejidad de los datos.

Este fue el último modelo probado en nuestra implementación y en este caso, el set de datos utilizado contiene los valores originales de la ubicación de la compañía y de residencia de los empleados, los cuales indican países en lugar de continentes.

Adicionalmente, luego de hacer un primer ensayo se evaluó la importancia de las variables y se descartaron las menos importantes: `remote_ratio`, `company_size` y `employment_type`.

Para definir los parámetros a utilizar en el modelo se realiza un random search utilizando 5 particiones y 30 iteraciones. De esta manera se identifica el mejor modelo, el cual tiene los siguientes parámetros:

depth: 4, iterations: 230, l2_leaf_reg: 2.8, learning_rate: 0.08.

Con estos hiper parámetros el RMSE obtenido es de 56054 USD.

Utilizamos MLFlow en Databricks para evaluar otros set de parámetros y corroborar el resultado anterior y efectivamente el experimento “unleashed-squid-605” da el mejor resultado:

Experiments > CatBoostRegression

metrics.rmse < 1 and params.model = "tree" Time created State: Active Datasets Sort: Created Columns Group by

Run Name	Created	Dataset	Duration	Source	Models	Metrics	Parameters
						rmse	depth iterations l2_leaf_reg learning_rate verbose
awesome-pig-683	4 days ago	-	5.7s	Modelof...	catboost	56105.25399...	3 150 3 0.25 0
bright-bear-962	4 days ago	-	8.0s	Modelof...	catboost	56171.43091...	6 443 1.4 0.04 0
unleashed-squid-605	4 days ago	-	6.1s	Modelof...	catboost	56054.07140...	4 230 2.8 0.08 0

2.2 Clasificación:

Escogimos utilizar RandomForest, CatBoost, LightGBM y XGBoost por la facilidad que presentan en el manejo de variables predictoras de tipo categórico, sin embargo a continuación se explicará únicamente los modelo de CatBoost y Random Forest que son los modelos que se subieron en DataBricks para realizar los experimentos

Como se mencionó anteriormente, decidimos implementar dos opciones de clases, una por rangos y otra por deciles(dado que las métricas obtenidas fueron demasiado bajas se hizo en terciles). En ambos casos, la métrica de evaluación que utilizamos es el Accuracy.

CatBoost: en el caso de Catboost para clasificación los parámetros usados fueron los mismos que en regresión y adicionalmente se agrega:

- Loss Function: Específica como el modelo evalúa los errores que ocurren durante el entrenamiento. Dado que es un ejercicio donde se tienen entre 3 y 4 clases se usó la función de pérdida MultiClass.

En las imágenes a continuación se puede observar los resultados obtenidos en los experimentos realizados en Databricks usando el modelo explicado anteriormente y en los experimentos con 4 y 3 clases, a partir de estos resultados se observa que para el caso de 4 clases el mejor accuracy obtenido es 0.427 en 4 de los 5 experimentos realizados mientras que en el caso de 3 clases el mejor desempeño obtenido es de 0.51 en 2 de los 3 experimentos realizados en este caso, lo que evidencia que no existe una diferencia significativa al variar los valores de los parámetros sin importar si se aumenta por ejemplo el número de iteraciones para que el modelo logre capturar mejor las complejidades de los datos, por otro lado al comparar los resultados obtenidos usando 4 o 3 clases se puede observar que con 3 clases es donde se obtiene un mejor desempeño.

Random Forest JK

metrics.rmse < 1 and params.model = "tree" Time created State: Active Datasets Sort: accuracy Columns Group by

Run Name	Created	Dataset	Duration	Source	Models	Metrics	Parameters
						accuracy	max_feat maxdepth num_trees
defiant-sheep-709	5 days ago	-	8.6s	Random...	sklearn	0.42206342...	5 8 1000
worried-elk-405	5 days ago	-	5.0s	Random...	sklearn	0.42161679...	5 8 200
dashing-moose-3	5 days ago	-	9.4s	Random...	sklearn	0.42117016...	5 10 1000
illustrious-toad-42	5 days ago	-	4.6s	Random...	sklearn	0.41491737...	5 10 100
amusing-lamb-748	5 days ago	-	9.4s	Random...	sklearn	0.41447074...	5 15 1000

CatClassifier (target en terciles) ABC

metrics.rmse < 1 and params.model = "tree" Time created State: Active Datasets Sort: iterations Columns Group by

Run Name	Created	Dataset	Duration	Source	Models	Metrics	Parameters
						accuracy depth iterations l2_leaf_reg learning_rate	
overjoyed-crow-191	4 days ago	-	7.1s	Classifi...	catboost	0.5002233...	4 100 1 0.01
persistent-loon-695	4 days ago	-	7.0s	Classifi...	catboost	0.5140687...	4 200 5 0.1
defiant-kit-446	4 days ago	-	7.2s	Classifi...	catboost	0.5140687...	4 230 9.6 0.2

Random Forest: Este algoritmo de machine learning comúnmente usado de ML que combina los resultados obtenidos de los diferentes árboles de decisión para llegar a un resultado final algunas de las ventajas de este modelo son que es un modelo fácil de usar, robusto, y reduce los riesgo de sobreajuste al promediar los resultados de los árboles. [4]

Los parámetros usados en este modelo fueron:

- **Max_features:** Controla el número máximo de características a considerar cuando se está buscando la mejor división en cada nodo del árbol, modificar este parámetro puede afectar la capacidad del modelo de capturar la complejidad de los datos y la diversidad de los árboles
- **Min_samples_leaf:** Determina el número mínimo de muestras que deben haber en una hoja, en donde un número muy pequeño de muestras puede generar sobreajuste
- **Min_samples_split:** Determina el número mínimo de muestras necesarias para dividir un nodo del árbol de decisión, es decir este parámetro ayuda a controlar el sobreajuste y la complejidad de los árboles
- **Bootstrap:** Define si los árboles se entrenan usando el muestreo con reemplazo, es decir se seleccionan aleatoriamente subconjuntos de los datos para entre los árboles de decisión y se define si estos subconjunto se repiten o no
- **N_estimators, Max_depth:** Estos parámetros ya fueron explicados en la sección de los modelos de regresión

En las imágenes a continuación se puede observar los resultados obtenidos en los experimentos realizados en Databricks. Sin embargo, similar a los resultados obtenidos con CatBoost el desempeño en los experimentos de 4 Clases es de alrededor del 42% y en el de 3 clases es de alrededor del 50%, lo que nos confirma que no existe una diferencia significativa al variar los valores de los parámetros, incluso cuando se prueban con diferentes modelos

Random Forest JK

Add Description

Q metrics.rmse < 1 and params.model = "tree"

Time created

State: Active

Datasets

Sort: accuracy

Columns





















Group by

Table

Chart

Evaluation

Preview

	Run Name	Created	Dataset	Duration	Source	Models	Metrics	Parameters		
							accuracy	max_feat	maxdepth	num_trees
<input type="checkbox"/>	 defiant-sheep-709	 5 days ago	-	8.6s	 Random...	 sklearn	0.42206342...	5	8	1000
<input type="checkbox"/>	 scorned-elk-405	 5 days ago	-	5.0s	 Random...	 sklearn	0.42161679...	5	8	200
<input type="checkbox"/>	 dashing-moose-3	 5 days ago	-	9.4s	 Random...	 sklearn	0.42117016...	5	10	1000
<input type="checkbox"/>	 fluttrious-toad-42	 5 days ago	-	4.6s	 Random...	 sklearn	0.41491737...	5	10	100
<input type="checkbox"/>	 amusing-lamb-748	 5 days ago	-	9.4s	 Random...	 sklearn	0.41447074...	5	15	1000

RFClassifier (target en terciles) ABC

Add Description

Permissions

Share

Metrics:rmse < 1 and params.model = "tree"

Time created

State: Active

Datasets

Sort: min_samples_split

Columns

Group by

+ New run

Table

Chart

Evaluation

Preview

		Run Name	Created	Dataset	Duration	Source	Models	Metrics	Parameters					
								accuracy	bootstrap	max_depth	max_features	min_samples_l	min_samples_split	n_estimators
<input type="checkbox"/>		serious-hound-387	4 days ago	-	8.5s	RFClass...	sklearn	0.514068...	True	15	sqrt	4	9	201
<input type="checkbox"/>		zealous-goose-577	4 days ago	-	7.3s	RFClass...	sklearn	0.50334...	False	10	sqrt	4	5	150
<input type="checkbox"/>		intelligent-cod-640	4 days ago	-	6.6s	RFClass...	sklearn	0.504242...	False	5	sqrt	3	3	50
<input type="checkbox"/>		bald-swain-456	4 days ago	-	7.9s	RFClass...	sklearn	0.514068...	True	15	sqrt	4	10	200
<input type="checkbox"/>		traveling-vole-430	4 days ago	-	7.9s	RFClass...	sklearn	0.514068...	False	15	sqrt	4	10	200

3. Observaciones y conclusiones sobre los modelos (Modificado en esta entrega)

3.1. Regresión:

Para regresión se experimenta con XGBoost, LightGradientBM y CatBoost, todos estos son modelos populares para problemas tanto de regresión como clasificación de los cuales ya nos extendimos anteriormente sobre sus ventajas y fortalezas. Son modelos conocidos en el ámbito del machine learning por sus buenos resultados y su optimización en la implementación sin embargo en el contexto de nuestro proyecto los resultados obtenidos no son óptimos. Incluso luego de probar con distintos hiper parámetros obtenemos RMSEs muy similares, todos con error (RMSE) en el entorno de 56.000 USD. Si comparamos este valor con el promedio de salario anual notamos que la proporción del error es del 36% del promedio de salario anual. Esto implica que las predicciones se desviarán en un +- 56.000 usd respecto al salario real.

3.2. Clasificación:

Luego de hacer las primeras implementaciones, inmediatamente notamos el bajo desempeño de los modelos en ambas opciones. La exactitud (accuracy) obtenida con la variable objetivo expresada en deciles, no supera el 20%, y como se mostró anteriormente, con 4 rangos fue del 40% y con tres del 50%.

Cabe anotar que hicimos experimentos eliminando las variables menos importantes, reduciendo el número de posiciones para las cuales se pretende hacer la predicción de salario, también se probaron modelos como SVM y Regresión Logística sin ningún éxito.

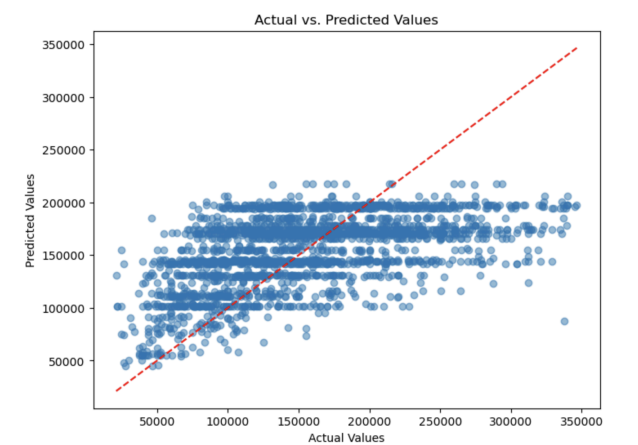
Por lo tanto consideramos que para el propósito de este proyecto y de acuerdo con los resultados obtenidos, un modelo de clasificación no es apropiado ya que el mejor modelo obtenido (3 clases) no logra explicar la expectativa de salarios que un empleador y empleado buscan para un tipo de trabajo en específico.

3.3 Conclusión final:

Consideramos que el hecho de que ninguno de los modelos implementados en Regresión y Clasificación resulte en un buen desempeño indica insuficiencia de información de valor en el set de datos.

Concluimos que el desempeño de los modelos está probablemente relacionado al hecho de que contamos con un número bajo de variables explicativas que en conjunto parecen no aportar significativamente a explicar las diferencias salariales y que no resultan suficiente para que los modelos puedan aprender de las mismas y generar predicciones acertadas.

En el siguiente gráfico se observa la pobre relación entre los valores predichos y los valores reales del conjunto de prueba. Como vemos están lejos de la predicción ideal (línea roja)

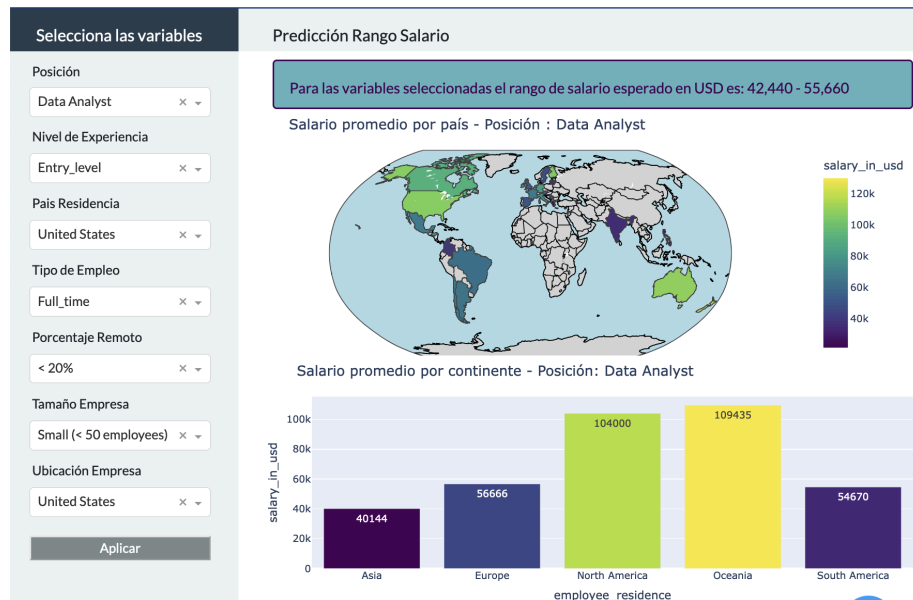


Sería interesante contar con mayor cantidad de variables referidas a las personas, cómo por ejemplo, edad, educación, género, así como contar con mayor granularidad respecto a las ubicaciones de las personas o empresas ya que el costo de vida al interior de los estados de EEUU (90.8% de los datos) varía y es de esperar que esto tenga un impacto en el salario ofrecido por las empresas.

Como lo mencionamos en la entrega anterior, por restricciones de tiempo decidimos continuar con el set de datos y modelo final elegido para dedicar los esfuerzos al despliegue de la solución, siendo este el objetivo principal del curso. Es así como para el desarrollo del tablero decidimos utilizar la implementación de CatBoost para regresión por dos motivos: 1) Como conclusión a la implementación de los modelos de clasificación consideramos que estos modelos no son los más apropiados para responder la pregunta de negocio y elegimos el enfoque de Regresión. 2) Dentro de las implementaciones realizadas para regresión es con CatBoost que se utiliza el set de datos con países en vez de continentes lo que creemos puede ser de mayor interés para los usuarios. Adicionalmente las diferencias en los RMSE entre los modelos desarrollados es insignificante (<1%) por lo que el RMSE no fue un determinante en la decisión.

4. Tablero Desarrollado

El tablero se construye con base en la maqueta propuesta en la primera entrega, en la cual se presentaba un tablero con tres páginas. Sin embargo, durante el desarrollo del mismo, nos dimos cuenta que es mejor presentar la información de las primeras dos en una sola página, como se muestra en la siguiente captura:



Los usuarios del tablero siguen siendo los mismos: individuos en busca de un cambio laboral y empresas o cazatalentos que quieren conocer los salarios en el mercado para las posiciones que buscan completar. Las preguntas que se responden con el tablero siguen siendo las mismas:

- 1) ¿Cuál es el rango potencial de salario esperado dadas las variables seleccionadas?
- 2) ¿Cómo se comparan los salarios para la posición en las distintas regiones?

Inputs: A través de los menús desplegables de la izquierda, los usuarios seleccionan la posición de interés, el nivel de experiencia, el país de residencia, el tipo de empleo, el % de tiempo remoto, el tamaño y ubicación de la empresa.

Outputs: Con base en información de entrada el modelo construido predice el salario esperado, el cual se presenta al usuario como un rango salarial al cual puede aspirar.

En cuanto a la tercera pregunta que se hizo en la maqueta ¿Cómo impactan las distintas variables en el salario correspondiente a la posición objetivo?, decidimos no incluirla en esta versión del proyecto dado que los modelos no tuvieron el desempeño que esperábamos.

El tablero se implementa utilizando Dash y el script se encuentra disponible en el repositorio: [Tablero Final](#). El script cambia con respecto a la segunda entrega, ya que ahora el despliegue se realiza en conjunto con la API, la cual se encarga de administrar el modelo, verificar los esquemas de entrada, hacer el preprocesamiento de las mismas y generar la predicción (salida). En el tablero se cargan las librerías y los datos para generar los gráficos descriptivos. Se definen y aplican los rangos de salarios que serán desplegados al usuario en la salida del tablero, con base en la predicción que se obtiene de la API. El dashboard actúa como interfaz para recibir las entradas seleccionadas por el usuario (callback), pasarlas a la API y después desplegar el rango de salarios con base en la predicción obtenida en la API (actualización de salida).

Nota Importante: Aquí decidimos recibir todas las variables que se utilizaron en la creación de los modelos, sin embargo en el modelo final sólo se utilizan cuatro, las más importantes (de acuerdo con el feature importance obtenido del modelo utilizado), en el siguiente orden: posición, nivel de experiencia, la ubicación de la empresa y en menor medida la ubicación del empleado. Se dejaron todas, pues en la medida que la diversidad de la información mejore, es posible que otras variables adquieran relevancia. Por ejemplo si se obtienen datos de más países y más posiciones.

Despliegue del tablero:

El tablero es desplegado en la nube haciendo uso de una instancia EC2. Este es el servicio de máquinas virtuales prestado por Amazon Web Services, y es un ejemplo de IaaS (infrastructure as a service). También hacemos uso de Docker a través de Railway. Esta última es una Platform as a Service que se encarga de instalar docker y generar la infraestructura necesaria para poder realizar el despliegue. Railway utiliza el Dockerfile incluido como parte del código para configurar el entorno que necesitamos en los contenedores de docker para correr la aplicación y nos proporciona la URL que los usuarios utilizarán.

En nuestro caso también utilizamos Railway tanto para el despliegue de la API así que tuvimos que crear una instancia adicional en EC2 y dos proyectos con sus respectivos servicios de despliegue asociados en Railway. Obtuvimos dos urls, una para la API, la cual se vincula en el tablero como variable de entorno para que cualquier persona pueda realizar su propio despliegue, y una url del tablero que es la que compartiremos con los usuarios para que accedan a la solución: <https://tranquil-balance-production.up.railway.app/>

5. Observaciones y conclusiones sobre el proyecto (Modificado en esta entrega)

Más allá de las conclusiones obtenidas sobre los modelos implementados y las oportunidades de mejora identificadas previamente, consideramos que este proyecto ha representado una valiosa experiencia de aprendizaje y colaboración en equipo. A lo largo de su desarrollo, tuvimos la oportunidad de explorar e implementar diversas herramientas clave para el despliegue de una solución basada en datos, lo que nos permitió consolidar conocimientos y habilidades prácticas.

En nuestras trayectorias laborales previas, hemos estado en contacto con algunas de las tareas relacionadas, pero no habíamos tenido la oportunidad de abordar un proceso completo *End-to-End*. Esta implementación nos brindó una visión integral, conectando los resultados finales con los requisitos de infraestructura y gestión de datos necesarios para hacer viable una solución de este tipo.

Evaluamos el proyecto como exitoso, ya que logramos trabajar de manera colaborativa, complementando nuestras fortalezas y apoyándonos mutuamente en áreas de mejora. Como resultado, desarrollamos un tablero funcional que cumple con los objetivos de negocio establecidos inicialmente. Asimismo, creemos que la solución desarrollada tiene un alto potencial de impacto y, con futuras mejoras, podría evolucionar hacia un producto altamente atractivo y competitivo. Entre las mejoras propuestas, destacamos:

1. **Ampliación del conjunto de datos:** Consideramos relevante incorporar un mayor número de características y una mayor diversidad en los datos. Actualmente, el dataset utilizado carece de variables relacionadas con las personas, tales como la educación, edad, género, y está principalmente enfocado en datos de Estados Unidos (que además no desglosa información a nivel estatal). Incorporar datos específicos de otros países podría mejorar significativamente la representatividad y precisión del modelo, ya que los salarios para un mismo puesto pueden variar considerablemente entre ellos. De esta manera, la integración de datos más diversos podría enriquecer los resultados de los modelos.
2. **Mejoras en el tablero de visualización:** Aunque el tablero desarrollado demostró ser intuitivo y atractivo para los usuarios, identificamos oportunidades para agregar funcionalidades adicionales. Por ejemplo, una vez se logre un modelo con alto poder predictivo, sería de gran utilidad contar con una herramienta que permita analizar el impacto de distintas variables sobre el salario asociado a una posición específica podría generar *insights* relevantes. Esto podría ayudar a los usuarios a tomar decisiones informadas, como adquirir habilidades específicas o considerar una reubicación geográfica.

En conclusión, este proyecto no solo ha cumplido con los objetivos iniciales, sino que también ha evidenciado su potencial para ser escalado y optimizado. Con las mejoras sugeridas, estamos convencidos de que podría convertirse en una solución aún más robusta y valiosa para los usuarios finales.

Anexos

1) Guía de acceso a los entregables no incluidos en este documento:

- Video de máximo 5 minutos donde se presente el problema y su contexto, los modelos construidos: [Youtube link](#)
- Manual de usuario del tablero: [link a github](#)
- Manual de instalación del tablero: [link a github](#)
- Repositorios con todo el código: [link a github](#)
- Fuentes de los modelos desarrollados: [link a github](#)
- Fuentes del tablero y API desarrollados: [link a github](#)
- Pantallazos de experimentos registrados en MLflow en una máquina de AWS EC2 o en Databricks: [link a github](#)
- Evidencia de despliegue de la API y tablero: [link a github](#)

2) Reporte de trabajo en equipo

Para esta entrega final esta fue la división de tareas acordada:

Tareas	Responsable Principal	Colaborador
Comenzar a crear presentación para el video	Johanna	Paulina
Experimentos de modelo Catboost Regresión	Alejandra	Paulina
Actualización de repositorio DVC, Edición del video final	Juan Camilo	
Documentación de modelo Catboost en Regresión y manual de instalación	Paulina	Alejandra
Conclusión Proyecto y Despliegue en Documento final	Paulina	Alejandra
Testeo y ajustes de despliegue del tablero	Todos	Todos
Actualización de código API y DASH	Alejandra	Johanna
Mantenimiento repositorio GitHub	Juan Camilo Quintana	Todos

Reuniones de equipo llevadas a cabo:

Jueves 14/11: Se revisan entregables del proyecto final y se verifican dudas para llevar a la tutoría. Luego de participar en la tutoría se acuerda trabajar en paralelo de forma individual en el desarrollo de la api del proyecto.

Sábado 16/11: Trabajo en equipo de 8 am a 15 pm ajustando la api e implementando el despliegue. Se acuerda división de tareas sobre todos los entregables y se acuerda trabajar en paralelo de forma individual en el desarrollo del taller de semana 7 para comenzar a implementar dicho taller en el proyecto.

Martes 19/11: Testeo de despliegue de API del proyecto con todos los integrantes.

Jueves 21/11: Testeo de despliegue e integración de API y Tablero en Railway con todos los integrantes.

Viernes 22/11: Actualización de repositorio y chequeo de status de los entregables finales. Se acuerdan tareas para los próximos días.

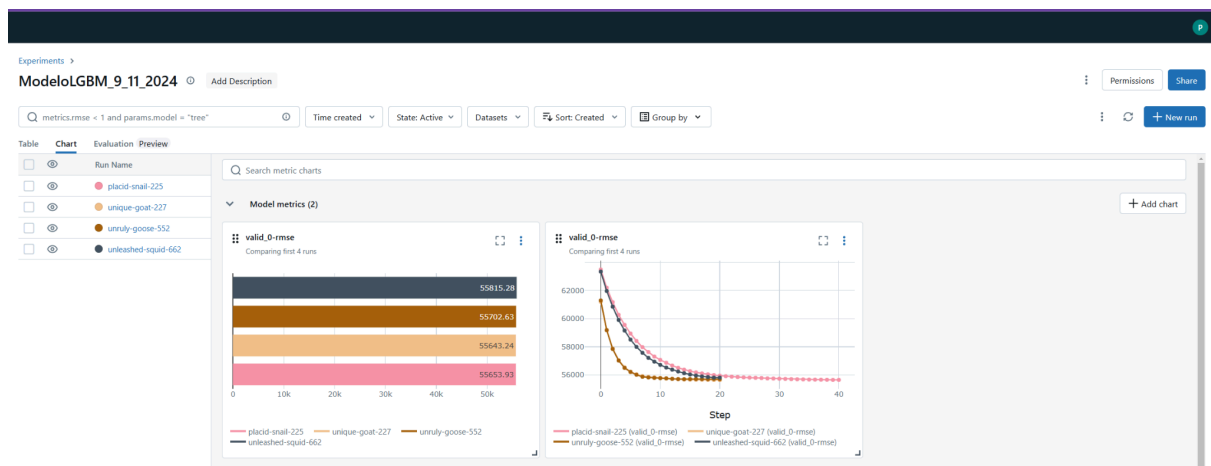
Lunes 25/11: Grabación del video y revisión de los entregables.

Martes 26/11: Revisión final de todos los entregables.

Como se mencionó en las conclusiones del proyecto el trabajo en equipo es lo que más resaltamos como positivo. Independientemente de lo que se infiera de los commits en Github, todos aportamos de igual manera al éxito del proyecto. Además fue una demostración de aprendizaje conjunto que esperamos poder replicar en el futuro.

3) Evidencia experimentos modelos Regresión:

LGBM

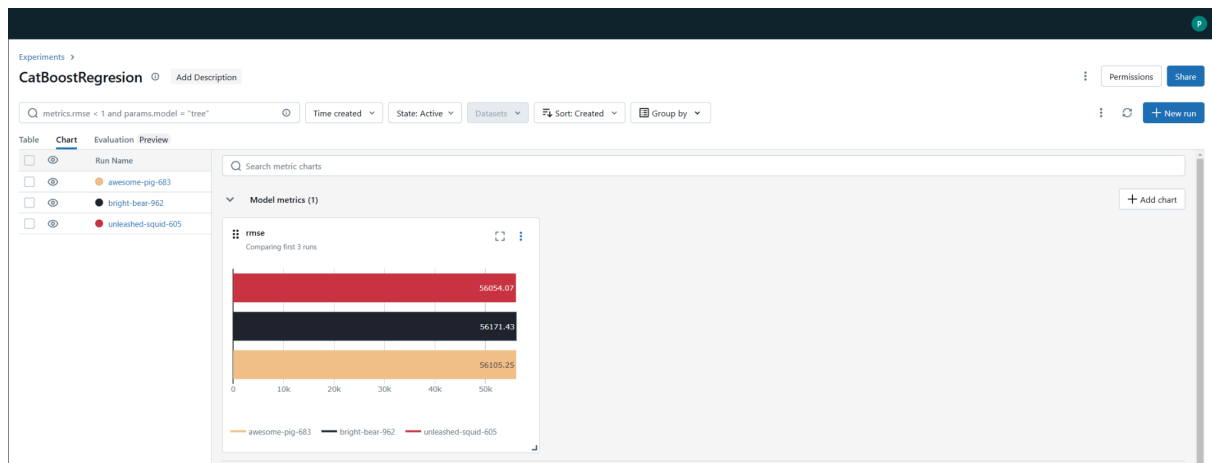


XGBoost

The screenshot shows the Databricks Experiments interface for an XGBoost model. The experiment is named 'XBGRegressor JCQ'. It displays a table of runs with columns for Run Name, Created, Dataset, Duration, Source, Models, Metrics, and Parameters. The table lists five runs with their names and metrics.

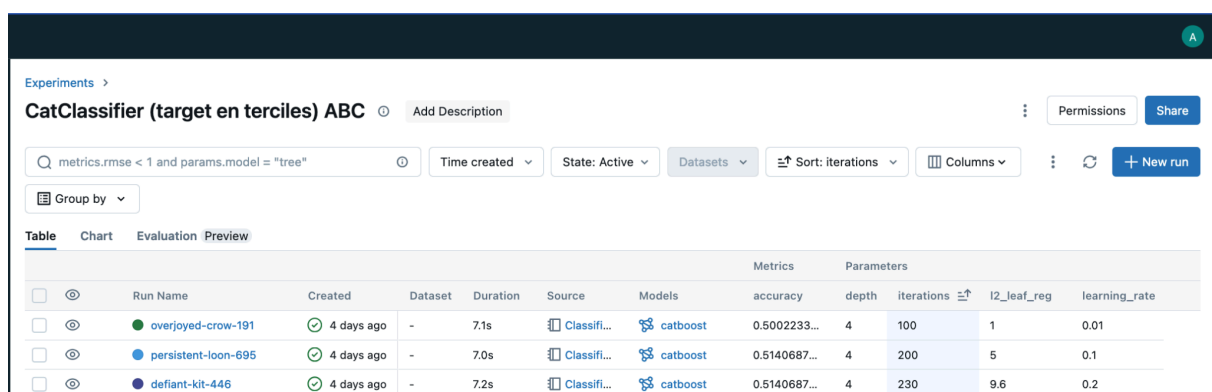
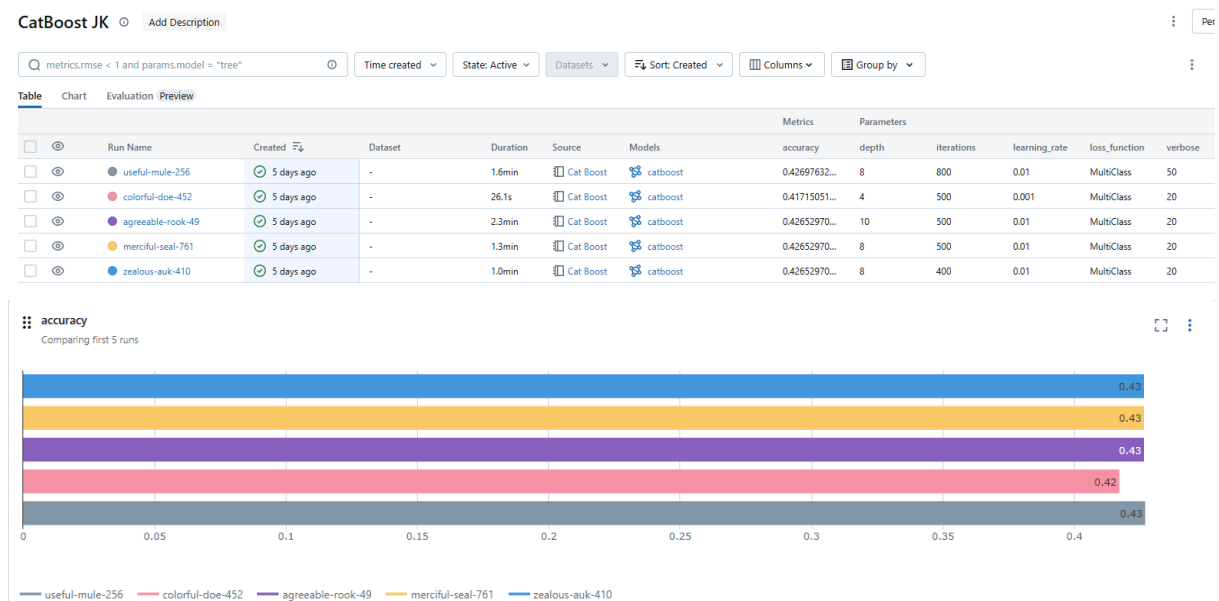
Run Name	Created	Dataset	Duration	Source	Models	Metrics	Parameters
casual-wolf-659	20 days ago	-	4.7s	Notebo...	sklearn	56020.90592...	colsample_bytree: 0.863493748..., learning_rate: 0.047932897..., max_depth: 3
languid-doe-822	20 days ago	-	5.6s	Notebo...	sklearn	56043.70550...	colsample_bytree: 0.648928654..., learning_rate: 0.051236450..., max_depth: 3
thundering-trout-936	20 days ago	-	4.6s	Notebo...	sklearn	56118.04231...	colsample_bytree: 0.863493748..., learning_rate: 0.05, max_depth: 4
fortunate-trout-750	20 days ago	-	5.4s	Notebo...	sklearn	56490.66313...	colsample_bytree: 0.863493748..., learning_rate: 0.3, max_depth: 10
serious-hawk-126	20 days ago	-	4.8s	Notebo...	sklearn	56494.05500...	colsample_bytree: 0.4, learning_rate: 0.3, max_depth: 10

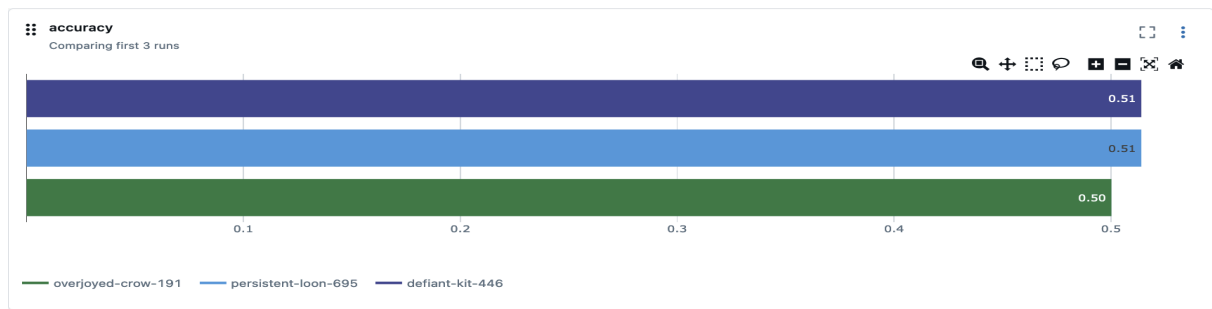
Catboost



4) Evidencia experimentos modelos Modelos Clasificación:

CatBoost





Random Forest:

Random Forest JK ⓘ Add Description

Q metrics.rmse < 1 and params.model = "tree" ⓘ

Time created ▾

State: Active ▾

Datasets ▾

Sort: accuracy ▾

Columns ▾

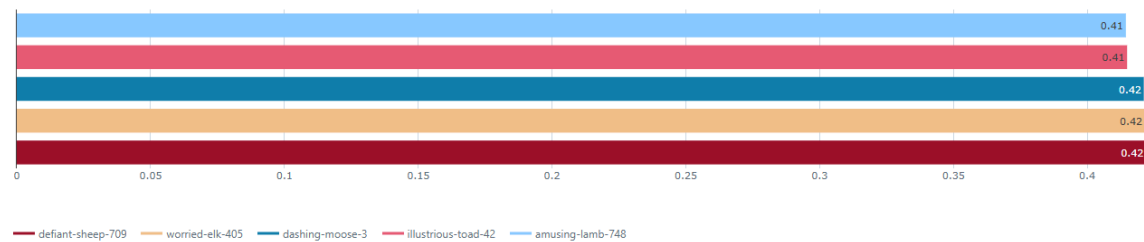
Group by ▾

Table Chart Evaluation **Preview**

		Run Name	Created	Dataset	Duration	Source	Models	Metrics		Parameters		
								accuracy		max_feat	maxdepth	num_trees
<input type="checkbox"/>		defiant-sheep-709	5 days ago	-	8.6s	Random...	sklearn	0.42206342...		5	8	1000
<input type="checkbox"/>		worried-elk-405	5 days ago	-	5.0s	Random...	sklearn	0.42161679...		5	8	200
<input type="checkbox"/>		dashing-moose-3	5 days ago	-	9.4s	Random...	sklearn	0.42117016...		5	10	1000
<input type="checkbox"/>		illustrious-toad-42	5 days ago	-	4.6s	Random...	sklearn	0.41491737...		5	10	100
<input type="checkbox"/>		amusing-lamb-748	5 days ago	-	9.4s	Random...	sklearn	0.41447074...		5	15	1000

accuracy

Comparing first 5 runs



Experiments >

RFClassifier (target en terciles) ABC ⓘ Add Description

Permissions Share

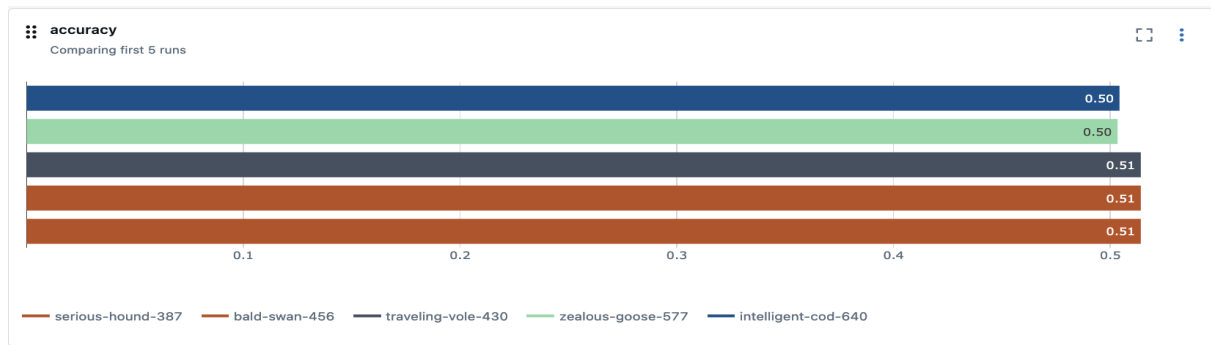
Q metrics.rmse < 1 and params.model = "tree" ⓘ

Time created ▾ State: Active ▾ Datasets ▾ Sort: min_samples_split ▾ Columns ▾

Group by ▾

Table Chart Evaluation **Preview**

		Run Name	Created	Dataset	Duration	Source	Models	Metrics		Parameters					
								accuracy		bootstrap	max_depth	max_features	min_samples_l	min_samples_spl	n_estimators
<input type="checkbox"/>		serious-hound-387	4 days ago	-	8.5s	RFClass...	sklearn	0.514068...		True	15	sqrt	4	9	201
<input type="checkbox"/>		zealous-goose-577	4 days ago	-	7.3s	RFClass...	sklearn	0.50334...		False	10	sqrt	4	5	150
<input type="checkbox"/>		intelligent-cod-640	4 days ago	-	6.6s	RFClass...	sklearn	0.504242...		False	5	sqrt	3	3	50
<input type="checkbox"/>		bald-swan-456	4 days ago	-	7.9s	RFClass...	sklearn	0.514068...		True	15	sqrt	4	10	200
<input type="checkbox"/>		traveling-vole-430	4 days ago	-	7.9s	RFClass...	sklearn	0.514068...		False	15	sqrt	4	10	200



5) Bibliografía

ArcGISPro. (2024). *Cómo funciona el algoritmo XGBoost*. Obtenido de:

<https://pro.arcgis.com/es/pro-app/latest/tool-reference/geoai/how-xgboost-works.htm#:~:text=B%C3%BAsqueda%20de%20divisiones%20sensibles%20a,el%20c%C3%A1lculo%20de%20la%20ganancia.>

[1]Gellért Nacsá (2023). Top 5 Advantages That CatBoost ML Brings to Your Data to Make it Purr. Obtenido de: <https://www.kdnuggets.com/2023/02/top-5-advantages-catboost-ml-brings-data-make-purr.html>

[2]CatBoost in Machine Learning: A Detailed Guide (2024). Obtenido de: <https://www.datacamp.com/tutorial/catboost>

[3] <https://catboost.ai/en/docs/references/training-parameters/>

[4] What is random forest? Obtenido de:

<https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.>

[5] How to Create a Dashboard with Dash and Plotly (Part 1: Layout) (2022). Obtenido de: <https://sakizo-blog.com/en/487/>

LightGBM (Light Gradient Boosting Machine) (2024). Obtenido de:

<https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/>