Proyecto #4

Santiago Hernandez

Ian Tomas Felipe Fonseca

Alejandro Molina Aranza

Juan Manuel Diaz Castaneda

Introducción a inteligencia artificial

a) A lo largo del proyecto se realizaron diferentes actividades, empezamos creando el repositorio donde posteriormente se irían actualizando las actividades a lo largo del mes, de primera medida se escogió un dataset optimo, el cual cumpliera con las características dadas por el profesor, dichos datos fueron agregados al repositorio, en formato csv, posteriormente se creó el EDA

completo del proyecto, incluyendo nulos, ceros, gráficos etc., limpiamos el dataset, quitamos variables irrelevantes y se corrieron diferentes algoritmos de búsqueda, dependiendo de cada estudiante, se normalizaron los datos y se separaron en diferentes capas, se realizaron las predicciones de los algoritmos y se sacaron las conclusiones.

b) En la etapa de experimentación se desarrollaron diferentes modelos en Python, implementados en Jupyter Notebook.
Cada integrante del grupo trabajó con un algoritmo distinto de aprendizaje automático, de acuerdo con la asignación dada.
Se realizaron pruebas con los modelos Gradient Boosting Regressor, AGNES (clustering jerárquico), KNN (k-nearest neighbors) y AdaBoost, todos aplicados sobre los mismos datos previamente preparados.

Adicionalmente, se realizaron experimentos complementarios con tres algoritmos extra, lo que permitió comparar los resultados y verificar la estabilidad de los modelos frente a diferentes enfoques de aprendizaje.
Todo el proceso se llevó a cabo en Python, empleando librerías como *scikit-learn*, *pandas* y *matplotlib* para el entrenamiento, análisis y visualización de los resultados.

c)

Se evaluaron varios algoritmos para predecir el tiempo de entrega de pedidos de comida. A continuación, los resultados:

1. Lasso Regression

Hiperparámetros: alpha=0.1, max_iter=2000, tol=0.001

R^2 en validación: 0.81

MSE en validación: 87.62

R^2 en test: 0.67

RMSE en test: 12.46 minutos

MSE en test: 155.21

2. Decision Tree Regressor

Hiperparámetros: max_depth=5, min_samples_split=5, min_samples_leaf=8

R^2 en validación: 0.70

RMSE en validación: 11.68 minutos

R^2 en test: 0.70

RMSE en test: 11.68 minutos

3. Gradient Boosting Regression

Hiperparámetros: n_estimators=200, learning_rate=0.1, max_depth=2

R^2 en entrenamiento: 0.90

R^2 en validación: 0.70

MSE en validación: 0.34

R^2 en test: 0.77

MSE en test: 0.21

4. AdaBoost Regressor

Hiperparámetros: n_estimators=1000, learning_rate=0.1, base_estimator=DecisionTreeRegressor(max_depth=5)

R^2 en test: 0.70

RMSE en test: 0.51

MAE en test: 0.40

5. K-Nearest Neighbors (KNN)

Hiperparámetros: n_neighbors=15, weights='distance', p=2 (Euclidean)

R^2 en validación: 0.69

RMSE en validación: 11.91 minutos

R^2 en test: 0.59

RMSE en test: 13.87 minutos

6. K-Means Clustering

Hiperparámetros: n_clusters=3, init='random', max_iter=300

Silhouette Score: 0.28

Resultado: agrupación poco efectiva

7. AGNES (Agglomerative Clustering) con Random Forest

Clusters: 3 grupos (491, 360, 149 registros)

Accuracy en test: 0.90

Precision promedio: 0.86

Recall promedio: 0.79

F1-score promedio: 0.82

Comparación de modelos: El mejor rendimiento en test lo obtuvo Gradient Boosting Regression con R^2=0.77 y MSE=0.21, seguido de Lasso Regression (R^2=0.67) y Decision Tree (R^2=0.70). KNN mostró un rendimiento inferior (R^2=0.59). K-Means no fue efectivo para este problema.Análisis de características:Del análisis exploratorio, la distancia (Distance_km) es la variable más relevante (correlación 0.78 con el tiempo de entrega), seguida del tiempo de preparación (correlación 0.31). La experiencia del repartidor mostró correlación negativa débil (-0.09).

d)

1. Rendimiento de los algoritmos: Gradient Boosting Regression fue el mejor modelo (R^2=0.77 en test), indicando que explica el 77% de la varianza. Lasso y Decision Tree mostraron rendimientos similares (R^2≈0.67-0.70), mientras que KNN fue inferior (R^2=0.59).2. Factores que influyen en el tiempo de entrega:La distancia es el factor más importante (correlación 0.78), seguida del tiempo de preparación (0.31). La experiencia del repartidor tiene un impacto menor. Las variables categóricas (clima, tráfico, tipo de vehículo) también influyen, aunque en menor medida.3. Limitaciones identificadas:

K-Means no fue efectivo (Silhouette=0.28), sugiriendo que los datos no forman grupos bien definidos.

KNN mostró menor rendimiento, posiblemente por la naturaleza de los datos.

Algunos modelos presentaron diferencias entre entrenamiento y validación, indicando posible sobreajuste.

4. Aplicabilidad del modelo: Con R^2=0.77, el modelo puede ser útil para estimar tiempos de entrega, aunque hay margen de mejora. El RMSE de aproximadamente 12 minutos sugiere que las predicciones tienen un error promedio razonable para el contexto.5. Recomendaciones:

Gradient Boosting es el modelo recomendado para producción.

La distancia debe ser considerada como variable principal en cualquier modelo futuro.

Se recomienda recopilar más datos y explorar técnicas de ensemble o deep learning para mejorar el rendimiento.

6. Impacto práctico: El modelo puede ayudar a mejorar la experiencia del usuario proporcionando estimaciones más precisas y a optimizar la asignación de repartidores según las condiciones del pedido.

e) Para mejorar los resultados que se obtuvieron en el experimento se pueden intentar varias cosas, primero, sería buena idea ajustar mejor los parámetros de cada modelo, por ejemplo, probar diferentes valores de profundidad en el árbol de decisión o cambiar el valor de alpha en Lasso para encontrar el equilibrio ideal entre precisión y generalización y de esta manera corroborar si el mejor algoritmo en verdad es el más eficiente de acuerdo a los parámetros de entrada, también se puede preparar mejor la información, normalizando o estandarizando los datos para que todas las variables tengan una escala parecida y el modelo no le dé más importancia a unas que a otras. Otra opción es probar con modelos más potentes, como Random Forest o Gradient Boosting, que suelen dar mejores resultados porque combinan varios modelos y aprenden de los errores de los anteriores, además, sería podríamos plantear el usar una validación cruzada, ya que ayuda a comprobar que el modelo realmente funciona bien con distintos conjuntos de datos y no solo con uno en específico, por último, se podría mejorar la selección o creación de variables buscando nuevas formas de representar la información para que el modelo tenga más datos útiles para aprender y así logre un desempeño más alto y sobre todo preciso.