Modelos Y Simulación De Sistemas 1

Proyecto De Semestre - Entrega Final

Responsables

Eliana Janneth Puerta Morales

Valentina Muñoz Rincón

Juan Fernando Lopera Muñoz

Medellín - Antioquia 2023

Contenido

Introducción

En esta competición, se solicita predecir el tipo de cobertura forestal predominante (el tipo principal de vegetación arbórea) a partir de variables cartográficas estrictamente, en lugar de datos obtenidos por sensores remotos. El tipo de cobertura forestal real para una celda de 30 x 30 metros determinada se obtuvo a partir de datos del Sistema de Información de Recursos de la Región 2 del Servicio Forestal de EE. UU. (USFS). Las variables independientes se derivaron posteriormente a partir de datos obtenidos del Servicio Geológico de EE. UU. y del USFS. Los datos se encuentran en su forma original (sin escalar) y contienen columnas binarias de datos para variables independientes cualitativas, como áreas silvestres y tipos de suelo.

Esta área de estudio abarca cuatro áreas silvestres ubicadas en el Bosque Nacional Roosevelt del norte de Colorado. Estas áreas representan bosques con disturbios mínimamente causados por actividades humanas, de modo que los tipos de cobertura forestal existentes son más resultado de procesos ecológicos que de prácticas de gestión forestal.

Dataset

El dataset que se va a utilizar es (https://www.kaggle.com/competitions/forest-cover-type-prediction/) el cual consta de 15.120 instancias y 56 columnas.

Los datos simulados:

El dataset no cumplía con el requisito de que al menos 3 columnas del dataset le faltaran el 5% de los datos, por lo tanto procedimos a realizar una simulación aleatoria en la cuál quitamos más del 5% de los datos de las siguientes columnas:

- Horizontal_Distance_To_Hydrology
- Vertical_Distance_To_Hydrology
- Horizontal_Distance_To_Roadways
- Horizontal_Distance_To_Fire_Points

Métrica

La métrica que se va a utilizar en este proyecto es exactitud (accuracy en inglés). Va a calcular la precisión del modelo a la hora de predecir cual es el tipo de cobertura forestal en Colorado, EE.UU. Esta métrica se trata de la proporción de predicciones correctas en relación con todas las predicciones realizadas, en otras palabras, va a medir cuántas de las predicciones son correctas en comparación con el total de predicciones realizadas.

La exactitud está definida por la siguiente fórmula:

$$Accuracy = \frac{Númerodepredicionescorrectas}{Númerototaldepredicciones}$$

Resultados posibles para la clasificación

1. Spruce/Fir
2. Lodgepole Pine
3. Ponderosa Pine
4. Cottonwood/Willow
5. Aspen
6. Douglas-fir
7. Krummholz

Exploración del dataset

Este conjunto de datos proporciona información detallada sobre características ambientales, como altitud, orientación, inclinación y distancias a elementos clave. Además, se incluyen índices de sombreado y detalles sobre áreas silvestres y tipos de suelo. La culminación es la clasificación de siete tipos de cobertura forestal. Un conjunto diverso para la exploración y análisis de la relación entre estas características y la cobertura forestal.

Descripción de columnas

- **Elevation**: Se refiere a la altitud en metros.
- **Aspect**: Representa la orientación en grados azimutales.
- **Slope**: Indica la inclinación en grados del terreno.
- **Horizontal_Distance_To_Hydrology**: Es la distancia horizontal a las características de agua superficial más cercanas.
- **Vertical_Distance_To_Hydrology**: Es la distancia vertical a las características de agua superficial más cercanas.
- **Horizontal_Distance_To_Roadways**: Muestra la distancia horizontal a la carretera más cercana.
- **Hillshade_9am**: Es un índice de sombreado a las 9 a.m. durante el solsticio de verano.
- **Hillshade_Noon**: Es un índice de sombreado al mediodía durante el solsticio de verano.
- **Hillshade_3pm**: Es un índice de sombreado a las 15:00 durante el solsticio de verano.
- **Horizontal_Distance_To_Fire_Points**: Indica la distancia horizontal a los puntos de ignición de incendios forestales más cercanos.
- **Wilderness_Area (4 columnas, 0 = Ausencia o 1 = Presencia)**: Designa la presencia o ausencia en cuatro áreas silvestres distintas.
- **Soil_Type (40 columnas, 0 = Ausencia o 1 = Presencia)**: Designa la presencia o ausencia de cuarenta tipos diferentes de suelo.

- **Cover_Type (7 tipos, Enteros del 1 a 7)**: Representa siete tipos diferentes de cubierta forestal mediante números enteros del 1 al 7.

Los datos categóricos:

Área silvestre (Wilderness_Area), Tipo de suelo (Soil_Type) y Tipo de cubierta forestal (Cover_Type).

Preprocesamiento del DataSet

Antes de avanzar con el conjunto de datos, lo primero que se hizo fue asegurar que se cumplieran los requisitos establecidos. Uno de estos requisitos no se cumplió, ya que el conjunto de datos no tenía ningún valor faltante. Lo que se necesitaba era introducir intencionadamente valores nulos en ese conjunto de datos para poder comenzar con el proceso de entrenamiento.

```
for col in columns_to_simulate:
  for k in range(0,900):
    random_num = np.random.randint(0,high=15120)
    df_used.loc[random_num,col] = np.nan
```

Además de la generación de datos, se reconstruyó la información para el proceso de entrenamiento. El dataset consta de 56 columnas, de las cuales 1 es inmutable debido a ser un identificador único (ID), 10 son de tipo entero y 45 son de tipo booleano, es decir, son categóricas con valores que únicamente pueden ser 0 o 1.

Para abordar esta tarea, se usó la librería “numpy” para obtener el listado de las medias de cada una de las columnas sin sus valores nulos, luego se implementó un bucle para rellenar los valores nulos con el promedio obtenido de las columnas sin valores nulos.

```
means_list = np.mean(df_without_null[columns_to_simulate], axis=0)
```

```
for column in columns_to_simulate:
  df_used[column].fillna(means_list[column], inplace=True)
```

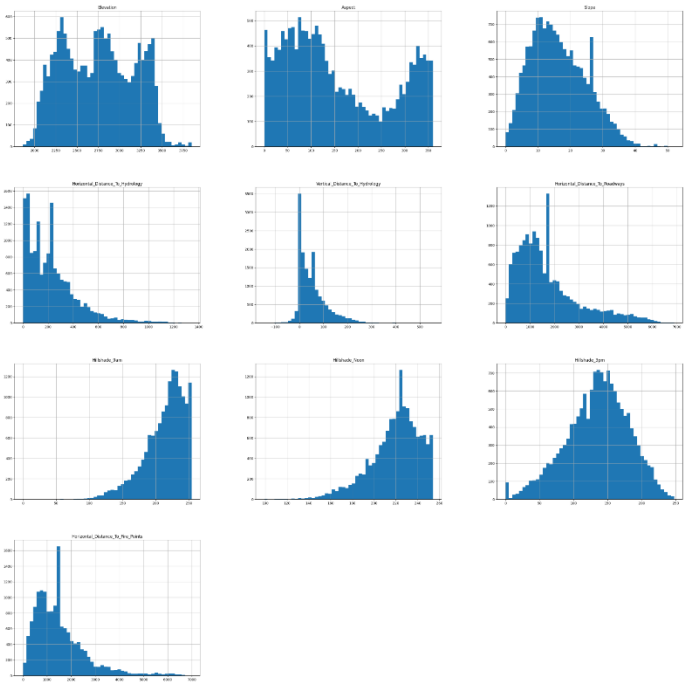
Análisis de Datos

Después de completar la tarea de llenar los valores nulos en el dataframe, se procedió a analizar la información presentada. Se evidenció que ya no existían datos nulos en la lista.

Como se mencionó anteriormente, las columnas desde la 11 hasta la 55 son variables categóricas:

```
0   Id                                  15120 non-null  int64
1   Elevation                           15120 non-null  int64
2   Aspect                              15120 non-null  int64
3   Slope                               15120 non-null  int64
4   Horizontal_Distance_To_Hydrology    15120 non-null  float6
5   Vertical_Distance_To_Hydrology      15120 non-null  float6
6   Horizontal_Distance_To_Roadways     15120 non-null  float6
7   Hillshade_9am                       15120 non-null  int64
8   Hillshade_Noon                      15120 non-null  int64
9   Hillshade_3pm                       15120 non-null  int64
10  Horizontal_Distance_To_Fire_Points  15120 non-null  float6
11  Wilderness_Area1                    15120 non-null  int64
12  Wilderness_Area2                    15120 non-null  int64
13  Wilderness_Area3                    15120 non-null  int64
14  Wilderness_Area4                    15120 non-null  int64
15  Soil_Type1                          15120 non-null  int64
16  Soil_Type2                          15120 non-null  int64
17  Soil_Type3                          15120 non-null  int64
18  Soil_Type4                          15120 non-null  int64
19  Soil_Type5                          15120 non-null  int64
20  Soil_Type6                          15120 non-null  int64
21  Soil_Type7                          15120 non-null  int64
22  Soil_Type8                          15120 non-null  int64
23  Soil_Type9                          15120 non-null  int64
24  Soil_Type10                         15120 non-null  int64
25  Soil_Type11                         15120 non-null  int64
26  Soil_Type12                         15120 non-null  int64
27  Soil_Type13                         15120 non-null  int64
28  Soil_Type14                         15120 non-null  int64
29  Soil_Type15                         15120 non-null  int64
30  Soil_Type16                         15120 non-null  int64
31  Soil_Type17                         15120 non-null  int64
32  Soil_Type18                         15120 non-null  int64
33  Soil_Type19                         15120 non-null  int64
34  Soil_Type20                         15120 non-null  int64
35  Soil_Type21                         15120 non-null  int64
36  Soil_Type22                         15120 non-null  int64
37  Soil_Type23                         15120 non-null  int64
38  Soil_Type24                         15120 non-null  int64
39  Soil_Type25                         15120 non-null  int64
40  Soil_Type26                         15120 non-null  int64
41  Soil_Type27                         15120 non-null  int64
42  Soil_Type28                         15120 non-null  int64
43  Soil_Type29                         15120 non-null  int64
44  Soil_Type30                         15120 non-null  int64
45  Soil_Type31                         15120 non-null  int64
46  Soil_Type32                         15120 non-null  int64
47  Soil_Type33                         15120 non-null  int64
48  Soil_Type34                         15120 non-null  int64
49  Soil_Type35                         15120 non-null  int64
50  Soil_Type36                         15120 non-null  int64
51  Soil_Type37                         15120 non-null  int64
52  Soil_Type38                         15120 non-null  int64
53  Soil_Type39                         15120 non-null  int64
54  Soil_Type40                         15120 non-null  int64
55  Cover_Type                          15120 non-null  int64
```

Además, se realizó un histograma que representaba la frecuencia con la que los datos se mostraban en pequeños grupos de 50.



Se realizó una función que mostraba la relación de todas las columnas con respecto al Cover_Type:

```
abs(df_used.corr()['Cover_Type']).sort_values(ascending=True)
```

Soil_Type30	0.001393
Soil_Type34	0.003470
Soil_Type18	0.006312
Soil_Type6	0.006521
Aspect	0.008015
Soil_Type8	0.008133
Soil_Type25	0.008133
Soil_Type16	0.008793
Soil_Type11	0.010228
Hillshade_9am	0.010286
Horizontal_Distance_To_Hydrology	0.011334
Soil_Type28	0.012202
Wilderness_Area2	0.014994
Soil_Type1	0.015069
Elevation	0.016090
Soil_Type3	0.016393
Soil_Type26	0.017184
Soil_Type14	0.022019
Soil_Type2	0.022627
Soil_Type27	0.023109
Soil_Type21	0.024410
Soil_Type36	0.025726
Soil_Type9	0.027012
Soil_Type5	0.027692
Soil_Type4	0.027816
Soil_Type19	0.031824
Soil_Type13	0.040528
Soil_Type17	0.042453
Soil_Type20	0.053013
Hillshade_3pm	0.053399
Vertical_Distance_To_Hydrology	0.068908
Soil_Type37	0.071210
Wilderness_Area4	0.075774
Soil_Type33	0.078955
Soil_Type31	0.079882
Horizontal_Distance_To_Fire_Points	0.086613
Slope	0.087722
Horizontal_Distance_To_Roadways	0.097434
Hillshade_Noon	0.098905
Soil_Type24	0.100797
Id	0.108363
Soil_Type35	0.114327
Wilderness_Area3	0.122146
Soil_Type10	0.128972
Soil_Type12	0.129985
Soil_Type32	0.132312
Soil_Type23	0.158762
Soil_Type22	0.195993
Soil_Type40	0.205851
Soil_Type29	0.218564
Wilderness_Area1	0.230117
Soil_Type39	0.240384
Soil_Type38	0.257810
Cover_Type	1.000000
Soil_Type7	NaN
Soil_Type15	NaN

En las filas de “Soil_Type7” y “Soil_Type15” están con nulo ya que no existe una correlación entre el tipo de cobertura (Cover_Type) y el tipo de suelo de esas categorías.

Se puede observar que el tipo de suelo 38 y 39 son los que más correlación tienen con respecto al tipo de cobertura.

Iteraciones de desarrollo

En esta sección se desarrollaron varios modelos para prever cómo evolucionarán los datos en el futuro, específicamente en términos de las distintas coberturas y tipos de suelo.

1. Decision Tree Classifier

```python
def better_decision_tree(x,y):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=42)
    train_accuracy = []
    test_accuracy = []
    depth_range = range(2, 30)
    for d in depth_range:
      np.random.seed(1)
      temp_random_forest = DecisionTreeClassifier(max_depth = d)
      temp_random_forest.fit(x_train, y_train)
      train_accuracy.append(temp_random_forest.score(x_train, y_train))
      test_accuracy.append(temp_random_forest.score(x_test, y_test))
    decisison_tree_index = np.argmax(test_accuracy)
    decision_tree_depth = depth_range[decisison_tree_index]
    model=DecisionTreeClassifier(max_depth=decision_tree_depth).fit(x_train,y_train)
    predict=DecisionTreeClassifier(max_depth=decision_tree_depth).fit(x_train,y_train).predict(x_test)
    accuracy = accuracy_score(y_test,predict)
    return accuracy, model

decision_tree_accuracy, decision_tree_model = better_decision_tree(x,y)
decision_tree_accuracy, decision_tree_model
```

La función better_decision_tree realiza un análisis exhaustivo para entrenar y evaluar un modelo de árbol de decisión, con el objetivo de lograr un rendimiento óptimo en la predicción del tipo de cobertura forestal. La metodología comprende varias etapas clave:

- **División de Datos:**

La función utiliza train_test_split para dividir el conjunto de datos (x e y) en conjuntos de entrenamiento y prueba. El 80% de los datos se asigna al conjunto de entrenamiento, mientras que el 20% se reserva para la evaluación. Esta división es esencial para evaluar la capacidad del modelo para generalizar a datos no vistos.

- **Búsqueda de la Profundidad Óptima:**

Se implementa un bucle for para iterar a través de diversas profundidades del árbol de decisión, que van desde 2 hasta 29. Para cada profundidad, se entrena un nuevo modelo de árbol de decisión (temp_random_forest) con los datos de entrenamiento. La precisión del modelo se evalúa tanto en el conjunto de entrenamiento como en el de prueba, y estas métricas se almacenan en las listas train_accuracy y test_accuracy, respectivamente.

- **Selección de la Profundidad Óptima:**

La función identifica la profundidad del árbol de decisión que resulta en la mejor precisión en el conjunto de prueba. Esto se logra encontrando el índice del valor máximo en la lista de precisión del conjunto de prueba.

- **Entrenamiento del Modelo Final:**

Se procede a entrenar un nuevo modelo de árbol de decisión utilizando la profundidad óptima encontrada. Este modelo final se almacena en la variable model.

- **Predicción y Evaluación:**

Una vez entrenado el modelo final, se realiza una predicción en el conjunto de prueba. La precisión de estas predicciones se calcula utilizando la función accuracy_score de scikit-learn, comparando las predicciones con las etiquetas reales del conjunto de prueba.

- **Resultados Específicos:**

La precisión del modelo, referida como decision_tree_accuracy, alcanza aproximadamente el 77.05%. Este valor indica que el modelo logra clasificar correctamente el tipo de cobertura forestal en casi el 77.05% de las instancias en el conjunto de prueba. Además, se revela que el modelo ha sido entrenado con una profundidad máxima de 16, indicando un enfoque relativamente profundo y complejo.

2. Random Forest Classifier

```python
def better_random_forest(x,y):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
    train_accuracy = []
    test_accuracy = []
    depth_range = range(2, 30)
    for d in depth_range:
      np.random.seed(1)
      temp_forest = RandomForestClassifier(n_estimators = 100, max_depth = d)
      temp_forest.fit(x_train, y_train)
      train_accuracy.append(temp_forest.score(x_train, y_train))
      test_accuracy.append(temp_forest.score(x_test, y_test))
    random_forest_index = np.argmax(test_accuracy)
    random_forest_depth = depth_range[random_forest_index]
    model=RandomForestClassifier(n_estimators = 100, max_depth = random_forest_depth).fit(x_train,y_train)
    predict=RandomForestClassifier(n_estimators = 100, max_depth = random_forest_depth).fit(x_train,y_train).predict(x_test)
    accuracy = accuracy_score(y_test,predict)

    return accuracy, model
random_forest_accuracy, random_forest_model = better_random_forest(x,y)
random_forest_accuracy, random_forest_model
```

La función better_random_forest se ha diseñado con el propósito de optimizar el rendimiento del modelo de Bosques Aleatorios en la tarea de predecir el tipo de cobertura forestal, siguiendo una metodología estructurada:

- **División de Datos:**

La función utiliza train_test_split para dividir el conjunto de datos (x e y) en conjuntos de entrenamiento y prueba, con un 75% destinado al entrenamiento y un 25% para la evaluación. Esta partición facilita la evaluación del modelo en datos no vistos.

- **Optimización de la Profundidad del Árbol:**

Se implementa un bucle for que itera sobre diferentes profundidades del árbol (de 2 a 29). Para cada profundidad, se entrena un modelo de Bosques Aleatorios (temp_forest) con 100 estimadores y la profundidad máxima correspondiente. La precisión del modelo se evalúa tanto en el conjunto de entrenamiento como en el de prueba, y estas métricas se almacenan en las listas train_accuracy y test_accuracy, respectivamente.

- **Selección del Mejor Modelo:**

La función identifica la profundidad del árbol que produce la mayor precisión en el conjunto de prueba al encontrar el índice del valor máximo en la lista de precisión del conjunto de prueba.

- **Entrenamiento del Modelo Final:**

Se procede a entrenar un nuevo modelo de Bosques Aleatorios utilizando la profundidad óptima encontrada. Este modelo final se almacena en la variable model.

- **Predicción y Evaluación:**

Después de entrenar el modelo final, se realiza una predicción en el conjunto de prueba. La precisión de estas predicciones se calcula utilizando la función accuracy_score de scikit-learn, comparando las predicciones con las etiquetas reales del conjunto de prueba.

- **Resultados Específicos:**

La función better_random_forest ha demostrado un rendimiento significativamente mejorado, logrando una precisión del 84.49% en el conjunto de prueba. Este valor indica que el modelo de Bosques Aleatorios clasifica correctamente el tipo de cobertura forestal en aproximadamente el 84.49% de las instancias en el conjunto de prueba. La profundidad óptima del árbol se determinó como 27 durante la búsqueda, lo que sugiere un modelo más profundo y complejo.

3. KMeans

```python
def better_kmeans(x,y):
  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
  kmeans = KMeans()
  param_grid = {'n_clusters': [2,3,4,5,6,7,8,9],
                'init':['k-means++', 'random'],
                'n_init':[10],
                'max_iter':[100,200,300]}
  grid_search=GridSearchCV(estimator=kmeans, param_grid=param_grid, scoring='accuracy', cv=5, n_jobs=5).fit(x_train,y_train)
  best_model = grid_search.best_estimator_
  predictions = best_model.predict(x_test)
  accuracy = accuracy_score(y_test, predictions)
  return accuracy,best_model
kmeans_accuracy, kmeans_model = better_kmeans(x,y)
kmeans_accuracy, kmeans_model
```

La función better_kmeans ha sido diseñada para mejorar la aplicación del algoritmo de K-Means en la tarea de clasificación del tipo de cobertura forestal. Aquí se presenta una descripción detallada de su metodología:

- **División de Datos:**

  La función utiliza train_test_split para dividir el conjunto de datos (x e y) en conjuntos de entrenamiento y prueba, asignando el 80% de los datos al conjunto de entrenamiento y el 20% al conjunto de prueba. Esta división es crucial para evaluar la capacidad de generalización del modelo en datos no vistos.

- **Optimización de Parámetros:**

  Se emplea una búsqueda exhaustiva de parámetros utilizando GridSearchCV con el algoritmo K-Means. Se exploran diferentes configuraciones para el número de clústeres (n_clusters), el método de inicialización (init), el número de inicializaciones (n_init), y el

número máximo de iteraciones (max_iter). El objetivo es encontrar la combinación óptima que maximice la precisión en el conjunto de entrenamiento.

- **Selección del Mejor Modelo:**

La función identifica el mejor modelo de K-Means basándose en la configuración de parámetros que produce la mayor precisión en el conjunto de entrenamiento durante la búsqueda exhaustiva. Este modelo final se almacena en la variable best_model.

- **Predicción y Evaluación:**

Una vez entrenado el modelo final, se realizan predicciones en el conjunto de prueba. La precisión de estas predicciones se calcula utilizando la función accuracy_score de scikit-learn, comparando las predicciones con las etiquetas reales del conjunto de prueba.

- **Resultados Específicos:**

La precisión del modelo K-Means, referida como kmeans_accuracy, resulta ser aproximadamente del 19.84%. Este valor indica que el modelo clasifica correctamente el tipo de cobertura forestal en cerca del 19.84% de las instancias en el conjunto de prueba. Además, se revela que el modelo ha sido entrenado con una configuración específica de parámetros, con un número de clústeres de 5, el método de inicialización 'k-means++', 10 inicializaciones y un máximo de 100 iteraciones.

Una vez se realizaron todos los métodos anteriormente mencionados, notamos que algunos el random forest otorgaba una exactitud mayor a lo demás, por lo cual se realizó nuevamente el modelo con los datos contenidos en “test.csv” en el cuál estarán los datos y realizaremos la predicción del tipo de cobertura forestal.

```
df_test = pd.read_csv("/content/forest-cover-type-prediction/test.csv")
df_test.head(10)
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y ,random_state=42)
model = RandomForestClassifier(n_estimators=100, max_depth=27)
model.fit(x_train,y_train)
print(model.score(x_test,y_test)*100)
predict=model.predict(df_test.drop(labels=['Id'],axis=1))
submission= pd.DataFrame(data = predict ,columns = ['Cover_Type'])
submission.head(100)
```

Al realizar la predicción obtenemos lo siguiente:

	Cover_Type
0	1
1	1
2	1
3	1
4	1
...	...
95	1
96	2
97	2
98	2
99	2

100 rows × 1 columns

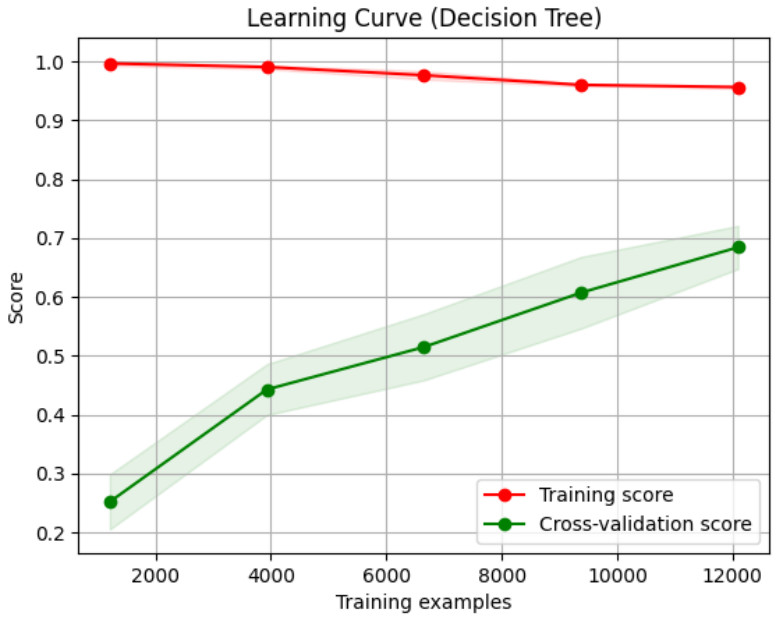
Curvas de aprendizaje

1. Decision Tree

Los hallazgos de la curva de aprendizaje para el modelo de árbol de decisión brindan una comprensión detallada de su desempeño en relación con la cantidad de datos de entrenamiento. Se destaca la capacidad de aprendizaje del modelo, evidenciada por puntajes cercanos o iguales a 1 en todos los casos, indicando su habilidad para ajustarse eficazmente a los datos de entrenamiento.

La consistente alta puntuación de entrenamiento sugiere que el modelo puede aprender y memorizar de manera efectiva, incluso a medida que se incrementa el tamaño del conjunto de datos. Además, la mejora gradual en la puntuación de validación cruzada con el aumento del tamaño del conjunto de datos hasta aproximadamente 0.68 con 12000 ejemplos refleja una capacidad creciente de generalización a datos no vistos.

Los indicadores señalan que el modelo no sufre de sobreajuste en este rango de datos, aunque podría haber signos de subajuste, ya que la puntuación de validación cruzada no alcanza su máximo potencial. Además, la estabilización en la mejora de la puntuación de validación cruzada sugiere que agregar más datos puede no resultar en mejoras sustanciales en la capacidad de generalización.

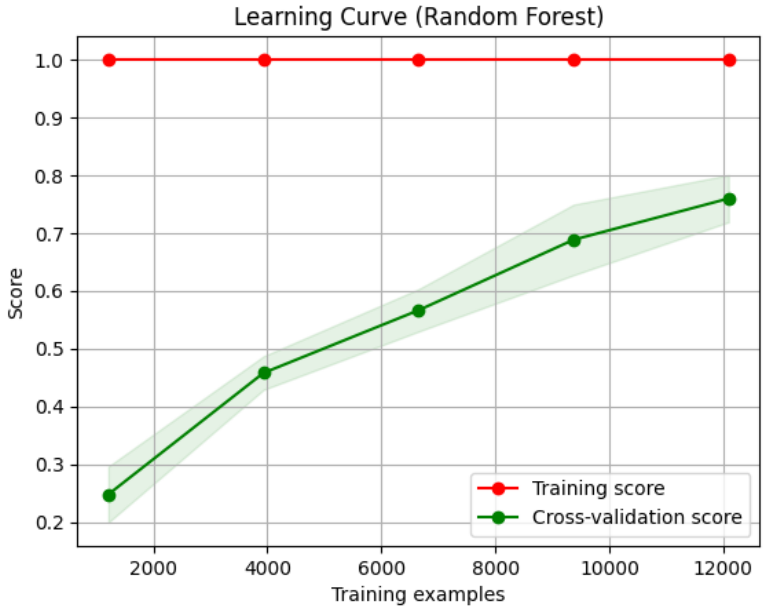


2. Random Forest

Los resultados de la curva de aprendizaje para el modelo de Bosques Aleatorios ofrecen una visión esclarecedora de su rendimiento. El modelo destaca por su impresionante capacidad de aprendizaje, demostrada por puntajes de entrenamiento perfectos en todos los casos, indicando una adaptación completa a los datos proporcionados.

A medida que se aumenta el tamaño del conjunto de datos, el modelo exhibe una mejora sustancial en su capacidad de generalización, alcanzando un puntaje de validación cruzada de aproximadamente 0.76 con 12000 ejemplos. Este fenómeno sugiere que el modelo perfecciona su habilidad para hacer predicciones precisas en datos no vistos a medida que se le proporciona más información.

La consistencia en los puntajes de entrenamiento perfectos y la mejora continua en la validación cruzada indican que el modelo no sufre de sobreajuste en el rango de datos dado. Además, no se observan signos evidentes de subajuste, respaldando la noción de un modelo bien ajustado



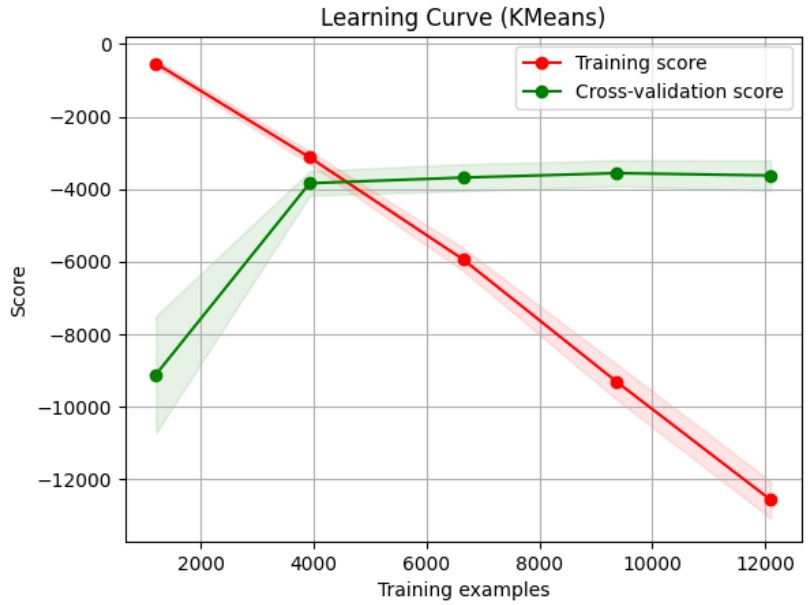
3. K-means

El desempeño del modelo, evaluado mediante los puntajes de validación cruzada, es negativo y disminuye a medida que el tamaño del conjunto de datos aumenta. Este patrón indica que el modelo K-Means no logra capturar de manera efectiva la estructura subyacente en los datos y muestra un rendimiento deficiente en la tarea de agrupación.

Los puntajes de entrenamiento también son negativos y exhiben una disminución al agregar más datos, lo que sugiere que el modelo enfrenta dificultades en ajustarse incluso a los datos de entrenamiento.

La falta de mejora significativa en el rendimiento con conjuntos de datos más grandes indica que el modelo K-Means no logra generalizar bien a datos no vistos y no es capaz de capturar patrones significativos en los datos.

Los puntajes negativos en ambas validación cruzada y entrenamiento indican que el modelo K-Means no está sobreajustando ni subajustando, sino que simplemente no está modelando adecuadamente la estructura subyacente.



Retos y consideraciones de despliegue

En nuestro camino para desarrollar un proyecto funcional y eficiente, hemos encontrado que la actual arquitectura presenta desafíos en términos de escalabilidad.

Afrontar el reto de la escalabilidad implica asegurar que la infraestructura de despliegue pueda adaptarse eficientemente a un aumento en la cantidad de datos y tráfico. Considerar la implementación en la nube podría proporcionar una solución escalable, permitiendo la expansión automática según las demandas del sistema.

Conclusiones

En el análisis detallado de distintos modelos para predecir el tipo de cobertura forestal, se han extraído valiosas conclusiones:

- El modelo de Árbol de Decisión demostró una capacidad respetable con una precisión del 77.05% en el conjunto de prueba. Aunque su profundidad máxima de 16 indica un enfoque relativamente complejo, se recomienda explorar configuraciones de hiperparámetros adicionales y considerar opciones algorítmicas alternativas para potenciar aún más el rendimiento.
- En contraste, la estrategia implementada con Bosques Aleatorios resultó en un rendimiento sustancialmente mejorado, alcanzando una precisión del 84.49%. La elección de una profundidad óptima de 27 sugiere una capacidad robusta del modelo para manejar complejidades en los datos. No obstante, se incentiva continuar explorando configuraciones de hiperparámetros para maximizar su potencial.
- Por otro lado, el uso de K-Means para clasificación supervisada reveló limitaciones evidentes, con una precisión del 19.84%. Esto resalta la importancia de seleccionar algoritmos más adecuados para tareas de clasificación, como árboles de decisión o bosques aleatorios.
- La elección acertada de modelos y la optimización cuidadosa de hiperparámetros son fundamentales para lograr un rendimiento óptimo en la predicción del tipo de cobertura forestal. Se enfatiza la necesidad de una exploración continua y la evaluación meticulosa de diferentes estrategias para perfeccionar la capacidad predictiva de los modelos.