Inteligencia Artificial Básico

Contenido

Inteligencia Artificial

Bootcamp: Inteligencia Artificial Básico

Eje de formación: Inteligencia Artificial
Nivel de formación: Básico
Duración total: 179 horas
Dedicación semanal: 14 horas por semana
Duración en meses: Aproximadamente 3 meses
Lenguajes de programación: Python

Competencias a Desarrollar

- Tratar conceptos fundamentales de la inteligencia artificial.
- Utilizar aplicaciones prácticas de la inteligencia artificial.
- Aplicar el proceso general para la construcción de modelos de machine learning.
- Conocer técnicas comunes de aprendizaje inductivo y supervisado.
- Identificar un diseño experimental adecuado para proyectos de machine learning.
- Construir chatbots y asistentes virtuales primarios.

Descripción del Bootcamp

Este bootcamp proporciona una introducción a los conceptos y aplicaciones fundamentales de la inteligencia artificial (IA). Los participantes aprenderán a aplicar técnicas básicas de machine learning, construir chatbots y asistentes virtuales, y desarrollar una comprensión sólida de los principios de la IA.

Kit programador requerido

- Computadora con acceso a internet
- Editor de código (Visual Studio Code, Jupyter Notebook)
- Python instalado
- Bibliotecas de Python para IA (scikit-learn, numpy, pandas, TensorFlow)

Justificación

El conocimiento de los fundamentos de la inteligencia artificial es crucial en el mundo moderno, donde la IA se está convirtiendo en una herramienta indispensable en diversas industrias. Este bootcamp proporciona una base sólida en IA, preparando a los participantes para roles iniciales en el desarrollo de aplicaciones de IA.

Rubrica de aprobación

Competencia	Criterio de Evaluación	Ponderación (%)
Fundamentos de IA	Comprensión de conceptos básicos	20
Aplicaciones prácticas de IA	Implementación de técnicas de machine learning	30
Construcción de modelos de ML	Desarrollo y evaluación de modelos	30
Proyecto final	Integración de todas las competencias en un proyecto	20

Estructura Curricular

Inicio

English code (lección 1) Jobs in Tech

English code (lección 2) System Software

English code (lección 3) Programming Software

English code (lección 4) Application Software

English code (lección 5) Programming Language Java – Java Scripts

Misión 1: Fundamentos de la Inteligencia Artificial

Mapa de Ruta - Misión 1: Fundamentos de la Inteligencia Artificial

Mapa de Ruta

Objetivo General: Proporcionar a los participantes una comprensión sólida de los conceptos fundamentales de la inteligencia artificial, incluyendo el aprendizaje supervisado e inductivo. Los estudiantes aprenderán a implementar algoritmos básicos de IA y desarrollar un proyecto práctico en un entorno colaborativo.

Objetivos del Bootcamp

1. **Introducción a la Inteligencia Artificial:**
   - Entender los principios básicos y la historia de la IA.
   - Familiarizarse con las aplicaciones y el impacto de la IA en diversos sectores.

2. **Aprendizaje Supervisado e Inductivo:**
   - Comprender las diferencias entre aprendizaje supervisado, no supervisado y por refuerzo.
   - Conocer las técnicas y algoritmos utilizados en el aprendizaje supervisado e inductivo.

3. **Implementación Práctica:**
   - Aplicar técnicas fundamentales de IA utilizando Python y bibliotecas populares.
   - Desarrollar e implementar algoritmos básicos de IA.
   - Realizar un proyecto práctico aplicando los conceptos aprendidos.

Habilidades Digitales a Desarrollar

1. **Fundamentos de IA:**
   - Conocimiento de los principios y la historia de la IA.
   - Familiaridad con las aplicaciones de la IA en diferentes sectores.

2. **Técnicas de Aprendizaje:**
   - Comprensión del aprendizaje supervisado e inductivo.
   - Habilidad para implementar algoritmos básicos de IA.

3. **Implementación en Python:**
   - Uso de bibliotecas de Python para IA (scikit-learn, numpy, pandas).
   - Desarrollo de proyectos prácticos en IA.

Contenidos a Abordar

1. **Conceptos Básicos de IA:**
   - Introducción a la inteligencia artificial.
   - Historia y evolución de la IA.
   - Aplicaciones de la IA en diferentes sectores.

2. **Tipos de Aprendizaje:**
   - Aprendizaje supervisado: definición, ejemplos y aplicaciones.
   - Aprendizaje no supervisado e inductivo: definición, ejemplos y aplicaciones.
   - Comparación entre los diferentes tipos de aprendizaje.

3. **Técnicas Fundamentales:**
   - Algoritmos básicos de IA (k-NN, regresión lineal, árboles de decisión).
   - Implementación de algoritmos en Python.
   - Evaluación y ajuste de modelos.

Propósitos Educativos del Bootcamp

- **Comprensión Sólida:** Proporcionar a los participantes una comprensión sólida de los fundamentos de la inteligencia artificial.
- **Habilidades Prácticas:** Desarrollar habilidades prácticas para implementar algoritmos básicos de IA.

- **Aplicación de Conocimientos:** Aplicar los conocimientos adquiridos en proyectos prácticos y colaborativos.

Alineación con el Mercado Laboral

El contenido del bootcamp está alineado con las necesidades del mercado laboral en el ámbito de la inteligencia artificial. Los estudiantes adquirirán competencias demandadas por la industria, incluyendo:

- Conocimientos fundamentales de IA y aprendizaje automático.
- Habilidades prácticas en la implementación de algoritmos de IA.
- Capacidad para desarrollar proyectos prácticos en IA.

Preparación para Desafíos Digitales

Los participantes estarán preparados para enfrentar desafíos digitales relacionados con la inteligencia artificial, incluyendo:

- Implementación de algoritmos básicos de IA.
- Desarrollo de proyectos prácticos aplicando técnicas de IA.
- Resolución de problemas utilizando herramientas y técnicas de IA.

Contenidos de la Misión 1

Entrenamiento

- **Mapa de Ruta:** Introducción a la inteligencia artificial, aprendizaje supervisado e inductivo.
  - **Objetivo:** Proporcionar una comprensión inicial de la IA y los diferentes tipos de aprendizaje.
  - **Habilidades Digitales:** Conocimiento de los conceptos fundamentales de IA, aprendizaje supervisado e inductivo.
  - **Contenidos:**
    - Historia y aplicaciones de la IA.
    - Diferencias entre aprendizaje supervisado, no supervisado y por refuerzo.
    - Técnicas y algoritmos básicos de IA.

- **Preparación (15 horas):** Conceptos básicos de IA, tipos de aprendizaje, y técnicas fundamentales.
  - ** Introducción a la IA, historia y aplicaciones.
  - ** Tipos de aprendizaje: supervisado, no supervisado y por refuerzo.
  - ** Técnicas fundamentales: algoritmos básicos de IA.
  - ** Herramientas y bibliotecas en Python para IA.

- **English Code (8 horas):** Terminología en inglés sobre IA, lectura de documentación técnica.
  - **Contenido:**
    - Vocabulario técnico en inteligencia artificial.

    - Lectura y comprensión de documentación técnica.
    - Redacción de comentarios y documentación en inglés.

Experiencia

- **Simulación (10 horas):** Implementación de algoritmos básicos de IA.
  - Configuración del entorno de trabajo y herramientas.
  - Implementación de un clasificador simple (por ejemplo, k-NN).
  - Implementación de un modelo de regresión.
  - Evaluación y ajuste de los modelos.

- **Cápsulas:** Videos y tutoriales sobre conceptos fundamentales de IA.
  - **Contenido:** Recursos multimedia que refuerzan los conceptos aprendidos en las sesiones teóricas y prácticas.

- **Zona de Recarga (5 horas):** Habilidades blandas como pensamiento crítico y resolución de problemas.
  - Pensamiento crítico en la resolución de problemas de IA.
  - Colaboración y trabajo en equipo en proyectos de IA.

Conexión

- **Prosumidores (3 horas):** Desarrollo de un pequeño proyecto de IA.
  - **Contenido:** Desarrollo de un proyecto sencillo que aplique los conceptos aprendidos (por ejemplo, un sistema de recomendación básico).

- **Co-creación (10 horas):** Proyecto colaborativo enfocado en un sector específico.
  - Definición del proyecto y planificación.
  - Desarrollo colaborativo del proyecto.
  - Presentación y discusión del proyecto.

- **Satélites (2 horas):** Networking y presentación del proyecto final.
  - Taller de presentación de proyectos y actividades de networking.

Preparación (lección 1) Introducción a la IA, Historia y Aplicaciones

Objetivos:
- Introducir los conceptos básicos de la inteligencia artificial.
- Explorar la historia y evolución de la IA.
- Comprender las aplicaciones de la IA en diversos sectores.

Contenidos:

1. **Introducción a la Inteligencia Artificial**
   - **Definición de IA:**
     - Qué es la inteligencia artificial.
     - Diferencia entre IA estrecha (ANI), IA general (AGI) y superinteligencia artificial (ASI).

- **Importancia de la IA:**
  - Impacto de la IA en la vida diaria y en diferentes industrias.
- **Ramas de la IA:**
  - Aprendizaje automático (Machine Learning).
  - Procesamiento del lenguaje natural (NLP).
  - Visión por computadora.
  - Robótica.

2. **Historia y Evolución de la IA**
- **Historia temprana:**
  - Primeros conceptos y estudios sobre IA.
  - Alan Turing y el Test de Turing.
- **Desarrollo de la IA:**
  - Primeros sistemas de IA y el "verano de la IA".
  - Inviernos de la IA: Períodos de desilusión y baja inversión.
  - Resurgimiento de la IA: Avances en hardware y algoritmos.
- **Hitos recientes:**
  - Desarrollo de redes neuronales profundas.
  - Avances en procesamiento de datos y big data.

3. **Aplicaciones de la IA en Diversos Sectores**
- **Salud:**
  - Diagnóstico médico asistido por IA.
  - Análisis de imágenes médicas.
  - Predicción de enfermedades.
- **Finanzas:**
  - Análisis predictivo y trading algorítmico.
  - Detección de fraudes.
  - Evaluación de riesgos de crédito.
- **Transporte:**
  - Vehículos autónomos.
  - Optimización de rutas y logística.
- **Marketing y Ventas:**
  - Sistemas de recomendación.
  - Análisis de comportamiento del consumidor.
  - Publicidad personalizada.
- **Educación:**
  - Tutores inteligentes.
  - Análisis de desempeño estudiantil.
  - Personalización del aprendizaje.
- **Otros Sectores:**
  - Agricultura (agricultura de precisión).
  - Manufactura (automatización de procesos).
  - Medio ambiente (modelado climático y predicciones).

Preparación (lección 2) Tipos de Aprendizaje: Supervisado, No Supervisado y Por Refuerzo

Objetivos:
- Comprender los diferentes tipos de aprendizaje en la inteligencia artificial.
- Explorar ejemplos y aplicaciones de cada tipo de aprendizaje.
- Diferenciar entre aprendizaje supervisado, no supervisado y por refuerzo.

Contenidos:

1. **Aprendizaje Supervisado**
   - **Definición:**
     - Qué es el aprendizaje supervisado.
     - Conceptos de entrada y salida etiquetada.
   - **Algoritmos Comunes:**
     - Regresión lineal y logística.
     - Máquinas de vectores de soporte (SVM).
     - Árboles de decisión y bosques aleatorios.
     - Redes neuronales.
   - **Ejemplos y Aplicaciones:**
     - Clasificación de correos electrónicos (spam vs. no spam).
     - Predicción de precios de viviendas.
     - Reconocimiento de imágenes.
     - Análisis de sentimientos en redes sociales.

2. **Aprendizaje No Supervisado**
   - **Definición:**
     - Qué es el aprendizaje no supervisado.
     - Conceptos de entrada sin etiquetas.
   - **Algoritmos Comunes:**
     - Agrupamiento (clustering) con k-means.
     - Análisis de componentes principales (PCA).
     - Modelos de mezcla gaussiana.
   - **Ejemplos y Aplicaciones:**
     - Segmentación de clientes.
     - Reducción de dimensionalidad.
     - Detección de anomalías.
     - Agrupamiento de documentos.

3. **Aprendizaje por Refuerzo**
   - **Definición:**
     - Qué es el aprendizaje por refuerzo.
     - Conceptos de agente, entorno, acciones, recompensas y políticas.
   - **Algoritmos Comunes:**
     - Q-learning.
     - SARSA (State-Action-Reward-State-Action).
     - Algoritmos de aprendizaje profundo por refuerzo (Deep Q-Networks).

- **Ejemplos y Aplicaciones:**
  - Juegos (ajedrez, Go).
  - Control de robots.
  - Sistemas de recomendación dinámicos.
  - Optimización de rutas en redes de tráfico.

Preparación (lección 3) Técnicas Fundamentales: Algoritmos Básicos de IA

Objetivos:
- Familiarizarse con los algoritmos básicos de la inteligencia artificial.
- Comprender los fundamentos matemáticos y estadísticos detrás de estos algoritmos.
- Implementar algoritmos básicos de IA utilizando Python y bibliotecas populares.

Contenidos:

1. **Algoritmos de Clasificación**
   - **Regresión Lineal:**
     - Conceptos y ecuaciones básicas.
     - Implementación en Python usando scikit-learn.
     ```python
     from sklearn.linear_model import LinearRegression
     X = [[1], [2], [3], [4]]
     y = [2, 3, 4, 5]
     model = LinearRegression().fit(X, y)
     predictions = model.predict(X)
     print(predictions)
     ```
   - **k-Nearest Neighbors (k-NN):**
     - Conceptos y funcionamiento.
     - Implementación en Python.
     ```python
     from sklearn.neighbors import KNeighborsClassifier
     X = [[0], [1], [2], [3]]
     y = [0, 0, 1, 1]
     model = KNeighborsClassifier(n_neighbors=3).fit(X, y)
     predictions = model.predict([[1.5]])
     print(predictions)
     ```

2. **Algoritmos de Regresión**
   - **Regresión Lineal:**
     - Conceptos y ecuaciones básicas.
     - Implementación en Python usando scikit-learn (similar al ejemplo anterior).
   - **Regresión Logística:**
     - Conceptos y ecuaciones básicas.

- Implementación en Python.
```python
from sklearn.linear_model import LogisticRegression
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
model = LogisticRegression().fit(X, y)
predictions = model.predict([[1.5]])
print(predictions)
```

3. **Árboles de Decisión**
   - **Conceptos:**
     - Estructura de un árbol de decisión.
     - Criterios de división (gini, entropía).
   - **Implementación en Python:**
     ```python
     from sklearn.tree import DecisionTreeClassifier
     X = [[0], [1], [2], [3]]
     y = [0, 0, 1, 1]
     model = DecisionTreeClassifier().fit(X, y)
     predictions = model.predict([[1.5]])
     print(predictions)
     ```

Preparación y Simulación (lección 4) Herramientas y Bibliotecas en Python para IA

Objetivos:
- Familiarizarse con las herramientas y bibliotecas populares de Python utilizadas en IA.
- Aprender a configurar y utilizar estas herramientas para implementar algoritmos de IA.
- Realizar tareas básicas de preprocesamiento de datos y visualización.

Contenidos:

1. **Introducción a las Bibliotecas de Python para IA**
   - **NumPy:** Manipulación de arreglos y operaciones matemáticas.
     ```python
     import numpy as np
     array = np.array([1, 2, 3])
     print(array * 2)
     ```
   - **Pandas:** Manipulación y análisis de datos.
     ```python
     import pandas as pd
     data = {'name': ['Alice', 'Bob', 'Charlie'], 'age': [25, 30, 35]}
     df = pd.DataFrame(data)
     print(df)
     ```

```
- **Matplotlib:** Visualización de datos.
  ```python
  import matplotlib.pyplot as plt
  plt.plot([1, 2, 3], [4, 5, 6])
  plt.show()
  ```

2. **Preprocesamiento de Datos**
   - **Limpieza de Datos:**
     - Identificación y manejo de valores faltantes.
     - Detección y eliminación de duplicados.
     ```python
     df.dropna(inplace=True)
     df.drop_duplicates(inplace=True)
     ```
   - **Normalización y Estandarización:**
     - Escalado de características.
     ```python
     from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     data_scaled = scaler.fit_transform(df[['age']])
     print(data_scaled)
     ```

3. **Implementación y Visualización**
   - **Implementación de

 un Pipeline Simple:**
     - Combinar varias etapas de preprocesamiento y modelado.
     ```python
     from sklearn.pipeline import Pipeline
     pipeline = Pipeline([
         ('scaler', StandardScaler()),
         ('model', LinearRegression())
     ])
     pipeline.fit(X, y)
     predictions = pipeline.predict(X)
     print(predictions)
     ```
   - **Visualización de Resultados:**
     - Creación de gráficos para representar los datos y los resultados.
     ```python
     plt.scatter(X, y, color='blue')
     plt.plot(X, predictions, color='red')
     plt.show()
     ```
```

Configuración del Entorno de Trabajo y Herramientas

Objetivos:
- Configurar el entorno de desarrollo para trabajar con Python y bibliotecas de IA.
- Familiarizarse con las herramientas y bibliotecas necesarias para la implementación de algoritmos de IA.

Contenidos:

1. **Instalación de Python y Bibliotecas Esenciales:**
   - **Instalación de Python:**
     - Descargar e instalar Python desde la página oficial (python.org).
     - Verificación de la instalación.
     ```bash
     python --version
     ```
   - **Instalación de Bibliotecas Esenciales:**
     - Uso de `pip` para instalar bibliotecas como NumPy, Pandas, Matplotlib, y Scikit-learn.
     ```bash
     pip install numpy pandas matplotlib scikit-learn
     ```

2. **Configuración de un Entorno de Desarrollo:**
   - **Visual Studio Code:**
     - Instalación de Visual Studio Code.
     - Instalación de extensiones relevantes (Python, Jupyter).
   - **Jupyter Notebook:**
     - Instalación de Jupyter Notebook.
     ```bash
     pip install notebook
     ```
     - Creación y ejecución de un notebook.
     ```bash
     jupyter notebook
     ```

3. **Verificación de la Configuración:**
   - Crear un script o notebook simple para verificar la configuración del entorno.
   ```python
   import numpy as np
   import pandas as pd
   import matplotlib.pyplot as plt
   from sklearn import datasets

   print("Libraries imported successfully!")
   ```

Simulación (lección 5) Implementación de un Clasificador Simple (k-NN)

Objetivos:
- Implementar y comprender el algoritmo k-Nearest Neighbors (k-NN).
- Evaluar el rendimiento del clasificador utilizando un conjunto de datos.

Contenidos:

1. **Introducción al Algoritmo k-Nearest Neighbors (k-NN):**
   - **Conceptos Básicos:**
     - Funcionamiento del algoritmo k-NN.
     - Parámetros importantes (número de vecinos `k`).
   - **Aplicaciones del k-NN:**
     - Reconocimiento de patrones.
     - Clasificación de textos y documentos.

2. **Carga y Exploración del Conjunto de Datos (Iris Dataset):**
   - **Carga del Conjunto de Datos:**
     ```python
     from sklearn.datasets import load_iris
     data = load_iris()
     X = data.data
     y = data.target
     print(data.DESCR)
     ```
   - **Exploración de los Datos:**
     - Descripción de las características y las etiquetas.
     - Visualización de las características utilizando gráficos.
     ```python
     import seaborn as sns
     df = pd.DataFrame(data.data, columns=data.feature_names)
     df['target'] = data.target
     sns.pairplot(df, hue='target')
     plt.show()
     ```

3. **Implementación del Algoritmo k-NN:**
   - **División de Datos en Conjuntos de Entrenamiento y Prueba:**
     ```python
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
     ```
   - **Entrenamiento del Modelo k-NN:**
     ```python
     from sklearn.neighbors import KNeighborsClassifier
     model = KNeighborsClassifier(n_neighbors=3)
     model.fit(X_train, y_train)
     ```

```
- **Evaluación del Modelo:**
  - Predicción en el conjunto de prueba.
  - Cálculo de la precisión del modelo.
  ```python
  y_pred = model.predict(X_test)
  from sklearn.metrics import accuracy_score
  accuracy = accuracy_score(y_test, y_pred)
  print(f'Accuracy: {accuracy}')
  ```

4. **Ajuste de Parámetros y Validación Cruzada:**
  - **Ajuste del Número de Vecinos (`k`):**
  ```python
  for k in range(1, 11):
      model = KNeighborsClassifier(n_neighbors=k)
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      accuracy = accuracy_score(y_test, y_pred)
      print(f'k={k}, Accuracy={accuracy}')
  ```
  - **Validación Cruzada:**
  ```python
  from sklearn.model_selection import cross_val_score
  scores = cross_val_score(KNeighborsClassifier(n_neighbors=3), X, y, cv=5)
  print(f'Cross-validation scores: {scores}')
  print(f'Mean cross-validation score: {scores.mean()}')
  ```

Simulación (lección 6) Implementación de un Modelo de Regresión

Objetivos:
- Implementar y comprender los modelos de regresión lineal y logística.
- Evaluar el rendimiento de los modelos utilizando conjuntos de datos reales.

Contenidos:

1. **Introducción a la Regresión Lineal:**
  - **Conceptos Básicos:**
    - Funcionamiento del modelo de regresión lineal.
    - Aplicaciones de la regresión lineal.
  - **Carga y Exploración del Conjunto de Datos (Boston Housing):**
  ```python
  from sklearn.datasets import load_boston
  data = load_boston()
  X = data.data
```
```

```
y = data.target
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y
print(df.head())
```

- **Implementación del Modelo de Regresión Lineal:**
```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X, y)
predictions = model.predict(X)
plt.scatter(y, predictions)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.show()
```
- **Evaluación del Modelo:**
```python
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y, predictions)
print(f'Mean Squared Error: {mse}')
```

2. **Introducción a la Regresión Logística:**
   - **Conceptos Básicos:**
     - Funcionamiento del modelo de regresión logística.
     - Aplicaciones de la regresión logística.
   - **Carga y Exploración del Conjunto de Datos (Breast Cancer):**
```python
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
X = data.data
y = data.target
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y
print(df.head())
```
   - **Implementación del Modelo de Regresión Logística:**
```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=10000)
model.fit(X, y)
predictions = model.predict(X)
plt.scatter(y, predictions)
plt.xlabel('Actual Classes')
plt.ylabel('Predicted Classes')
plt.show()
```

- **Evaluación del Modelo:**
  ```python
  from sklearn.metrics import classification_report
  report = classification_report(y, predictions)
  print(report)
  ```

Simulación y Prosumidor (lección 7) Evaluación y Ajuste de Modelos. Desarrollo de un Pequeño Proyecto de IA

Objetivos:
- Evaluar el rendimiento de los modelos de IA implementados.
- Ajustar los hiperparámetros para mejorar el rendimiento.

Contenidos:

1. **Evaluación de Modelos:**
   - **Métricas de Evaluación:**
     - Precisión, recall, F1-score para clasificación.
     - Mean Squared Error (MSE), R-squared para regresión.
   - **Uso de Métricas para Evaluar Modelos:**
     ```python
     from sklearn.metrics import precision_score, recall_score, f1_score
     precision = precision_score(y_test, y_pred)
     recall = recall_score(y_test, y_pred)
     f1 = f1_score(y_test, y_pred)
     print(f'Precision: {precision}, Recall: {recall}, F1-score: {f1}')
     ```

2. **Ajuste de Hiperparámetros:**
   - **Búsqueda de Hiperparámetros Óptimos:**
     ```python
     from sklearn.model_selection import GridSearchCV
     param_grid = {'n_neighbors': range(1, 11)}
     grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
     grid.fit(X, y)
     print(f'Best parameters: {grid.best_params_}')
     print(f'Best cross-validation score: {grid.best_score_}')
     ```
   - **Validación Cruzada:**
     - Validación cruzada y uso de diferentes conjuntos de datos para evaluar la robustez del modelo.

3. **Documentación y Presentación de Resultados:**
   - **Documentación del Proceso:**

    - Redacción de un informe detallado sobre la implementación, evaluación y ajuste de los modelos.
  - **Presentación de Resultados:**
    - Creación de gráficos y tablas para representar los resultados obtenidos.
    ```python
    import seaborn as sns
    sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
    plt.show()
    ```

Desarrollo de un Pequeño Proyecto de IA

Objetivos:
- Identificar un problema sencillo que pueda ser resuelto utilizando técnicas de inteligencia artificial.
- Desarrollar un modelo de IA para abordar el problema seleccionado.
- Evaluar y presentar los resultados del proyecto.

Contenidos:

1. Selección y Definición del Proyecto

Objetivo: Identificar un problema específico y definir el alcance del proyecto de IA.

1. **Brainstorming de Ideas:**
  - **Ejemplos de Proyectos Simples:**
    - Clasificación de correos electrónicos (spam vs. no spam).
    - Predicción de precios de viviendas.
    - Análisis de sentimientos en comentarios de redes sociales.
    - Sistema de recomendación simple (por ejemplo, recomendación de películas).

2. **Definición del Proyecto:**
  - **Título del Proyecto:** Clasificación de correos electrónicos como spam o no spam.
  - **Descripción del Problema:** Desarrollar un modelo de IA que clasifique correos electrónicos en dos categorías: spam y no spam.
  - **Objetivos del Proyecto:**
    - Recolectar y preparar un conjunto de datos de correos electrónicos.
    - Implementar un algoritmo de clasificación (por ejemplo, Naive Bayes).
    - Evaluar el rendimiento del modelo.

2. Desarrollo del Proyecto

Objetivo: Implementar el modelo de IA y realizar el análisis correspondiente.

1. **Recolección y Preparación de Datos:**
  - **Carga del Conjunto de Datos:**

    - Uso de un conjunto de datos de correos electrónicos (por ejemplo, Enron Email Dataset).
    ```python
    import pandas as pd
    data = pd.read_csv('emails.csv')
    print(data.head())
    ```
  - **Preprocesamiento de Datos:**
    - Limpieza de texto (eliminación de puntuación, conversión a minúsculas, eliminación de palabras irrelevantes).
    ```python
    import re
    from nltk.corpus import stopwords

    def preprocess_text(text):
        text = re.sub(r'\W', ' ', text)
        text = text.lower()
        text = re.sub(r'\s+', ' ', text)
        return text

    data['text'] = data['text'].apply(preprocess_text)
    stop_words = set(stopwords.words('english'))
    data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))
    print(data.head())
    ```

2. **Implementación del Modelo de Clasificación:**
  - **Vectorización de Texto:**
    - Conversión del texto en vectores numéricos utilizando TF-IDF.
    ```python
    from sklearn.feature_extraction.text import TfidfVectorizer
    vectorizer = TfidfVectorizer()
    X = vectorizer.fit_transform(data['text'])
    y = data['spam']
    ```
  - **Entrenamiento del Modelo:**
    - División de los datos en conjuntos de entrenamiento y prueba.
    - Entrenamiento de un modelo de Naive Bayes.
    ```python
    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import MultinomialNB

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = MultinomialNB()
    model.fit(X_train, y_train)
    ```
  - **Evaluación del Modelo:**

- Predicción en el conjunto de prueba y evaluación del rendimiento.
```python
y_pred = model.predict(X_test)
from sklearn.metrics import classification_report, accuracy_score
print(classification_report(y_test, y_pred))
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
```

3. **Ajuste del Modelo:**
   - **Ajuste de Parámetros:**
     - Exploración de diferentes parámetros para mejorar el rendimiento del modelo.
     ```python
     from sklearn.model_selection import GridSearchCV

     param_grid = {'alpha': [0.1, 0.5, 1.0]}
     grid_search = GridSearchCV(MultinomialNB(), param_grid, cv=5)
     grid_search.fit(X_train, y_train)
     print(f'Best parameters: {grid_search.best_params_}')
     ```

4. **Visualización y Documentación:**
   - **Creación de Gráficos:**
     - Visualización de los resultados del modelo.
     ```python
     import matplotlib.pyplot as plt
     from sklearn.metrics import plot_confusion_matrix

     plot_confusion_matrix(model, X_test, y_test)
     plt.show()
     ```
   - **Documentación del Proceso:**
     - Redacción de un informe detallado sobre el proyecto, incluyendo la recolección de datos, preprocesamiento, implementación del modelo y evaluación.

3. Presentación de Resultados

Objetivo: Presentar el proyecto final, los resultados obtenidos y las conclusiones.

1. **Preparación de la Presentación:**
   - **Estructura de la Presentación:**
     - Introducción al proyecto y definición del problema.
     - Descripción del conjunto de datos y el preprocesamiento realizado.
     - Implementación del modelo y resultados obtenidos.
     - Conclusiones y posibles mejoras futuras.
   - **Materiales de Apoyo:**
     - Diapositivas con gráficos y tablas.
     - Código fuente documentado.

2. **Presentación del Proyecto:**
   - **Exposición del Proyecto:**
     - Presentación del problema, metodología y resultados.
     - Discusión de los desafíos enfrentados y cómo se resolvieron.
   - **Sesión de Preguntas y Respuestas:**
     - Abrir espacio para preguntas y retroalimentación de los compañeros y facilitadores.

Cápsulas (lección 8)

Videos y tutoriales sobre conceptos fundamentales de IA

Links de recursos según competencia especificas curados (yudi)

Co-creación (lección 9) Definición del Proyecto y Planificación

Objetivos:
- Seleccionar un sector específico y definir el problema a resolver.
- Planificar el proyecto colaborativo, asignar roles y responsabilidades.
- Establecer metas y cronograma para el desarrollo del proyecto.

Contenidos:

1. **Selección del Sector Específico y Problema a Resolver**
   - **Brainstorming de Sectores y Problemas:**
     - Educación: Predicción de rendimiento estudiantil.
     - Salud: Diagnóstico de enfermedades a partir de imágenes médicas.
     - Finanzas: Detección de fraudes en transacciones bancarias.
     - Marketing: Análisis de sentimientos en redes sociales.
   - **Definición del Problema:**
     - **Sector Seleccionado:** Salud.
     - **Problema a Resolver:** Diagnóstico de enfermedades pulmonares a partir de imágenes de rayos X.
     - **Objetivo del Proyecto:** Desarrollar un modelo de IA que clasifique imágenes de rayos X en categorías de enfermedades pulmonares.

2. **Planificación del Proyecto**
   - **Definición del Alcance:**
     - Especificar qué se incluirá y qué no se incluirá en el proyecto.
     - Establecer criterios de éxito para el proyecto.
   - **Asignación de Roles y Responsabilidades:**
     - Project Manager: Coordinación y supervisión general.
     - Data Scientists: Recolección y preprocesamiento de datos.
     - Machine Learning Engineers: Desarrollo e implementación del modelo.
     - Data Analysts: Evaluación y análisis de resultados.
   - **Desarrollo de un Cronograma:**
     - Dividir el proyecto en fases y establecer plazos para cada fase.
     - Crear un calendario de reuniones y revisiones de progreso.

3. **Establecimiento de Metas y Resultados Esperados**
   - **Metas a Corto Plazo:**
     - Recolección y preprocesamiento de datos de imágenes de rayos X.
     - Implementación inicial de un modelo de clasificación.
   - **Metas a Mediano Plazo:**
     - Evaluación y ajuste del modelo.
     - Validación cruzada y prueba en datos no vistos.
   - **Metas a Largo Plazo:**
     - Presentación y documentación del proyecto.
     - Publicación de resultados y recomendaciones.

Co-creación (lección 10) Desarrollo Colaborativo del Proyecto

Objetivos:
- Recolectar y preprocesar los datos necesarios para el proyecto.
- Implementar y evaluar el modelo de IA seleccionado.
- Ajustar y optimizar el modelo para mejorar su rendimiento.

Contenidos:

1. **Recolección y Preprocesamiento de Datos**
   - **Recolección de Datos:**
     - Obtener un conjunto de datos de imágenes de rayos X de fuentes públicas (por ejemplo, Kaggle).
     ```python
     import os
     import pandas as pd

     data_dir = 'path_to_dataset'
     images = [os.path.join(data_dir, img) for img in os.listdir(data_dir)]
     labels = [1 if 'disease' in img else 0 for img in images]
     df = pd.DataFrame({'image_path': images, 'label': labels})
     print(df.head())
     ```
   - **Preprocesamiento de Imágenes:**
     - Redimensionamiento y normalización de imágenes.
     ```python
     from PIL import Image
     import numpy as np

     def preprocess_image(image_path):
         img = Image.open(image_path)
         img = img.resize((128, 128))
         img = np.array(img) / 255.0
         return img
     ```

```
df['image'] = df['image_path'].apply(preprocess_image)
print(df['image'].shape)
```

2. **Implementación del Modelo de Clasificación**
   - **Selección del Modelo:**
     - Uso de una red neuronal convolucional (CNN) para la clasificación de imágenes.
     ```python
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

     model = Sequential([
         Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
         MaxPooling2D((2, 2)),
         Conv2D(64, (3, 3), activation='relu'),
         MaxPooling2D((2, 2)),
         Flatten(),
         Dense(128, activation='relu'),
         Dense(1, activation='sigmoid')
     ])

     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
     print(model.summary())
     ```
   - **Entrenamiento del Modelo:**
     ```python
     from sklearn.model_selection import train_test_split

     X = np.stack(df['image'].values)
     y = df['label'].values
     X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

     model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
     ```

3. **Evaluación y Ajuste del Modelo **
   - **Evaluación del Modelo:**
     ```python
     loss, accuracy = model.evaluate(X_val, y_val)
     print(f'Validation Accuracy: {accuracy}')
     ```
   - **Ajuste de Hiperparámetros:**
     - Exploración de diferentes configuraciones de la red y parámetros de entrenamiento.
     ```python
     from tensorflow.keras.optimizers import Adam

     def build_model(learning_rate):
     ```

```
        model = Sequential([
            Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
            MaxPooling2D((2, 2)),
            Conv2D(64, (3, 3), activation='relu'),
            MaxPooling2D((2, 2)),
            Flatten(),
            Dense(128, activation='relu'),
            Dense(1, activation='sigmoid')
        ])
        model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
        return model

    model = build_model(learning_rate=0.001)
    model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
    ```
```

## Co-creación y Satelites (lección 11) Presentación, Discusión y Networking del Proyecto

**Objetivos:**
- Preparar y presentar los resultados del proyecto.
- Discutir los hallazgos y recibir retroalimentación.

**Contenidos:**

1. **Preparación de la Presentación**
   - **Estructura de la Presentación:**
     - Introducción al problema y objetivos del proyecto.
     - Descripción del conjunto de datos y el preprocesamiento realizado.
     - Implementación del modelo y resultados obtenidos.
     - Conclusiones y recomendaciones para futuros trabajos.
   - **Materiales de Apoyo:**
     - Diapositivas con gráficos y tablas.
     - Código fuente documentado y resultados visualizados.

2. **Presentación del Proyecto**
   - **Exposición del Proyecto:**
     - Presentación del problema, metodología y resultados.
     - Discusión de los desafíos enfrentados y cómo se resolvieron.
   - **Sesión de Preguntas y Respuestas:**
     - Abrir espacio para preguntas y retroalimentación de los compañeros y facilitadores.

3. **Discusión y Retroalimentación**
   - **Evaluación de Resultados:**
     - Análisis crítico de los resultados obtenidos y su significado.
```

- **Retroalimentación:**
  - Recepción de sugerencias y comentarios para mejorar el proyecto.
- **Planificación de Mejoras Futuras:**
  - Identificación de áreas de mejora y planificación de pasos futuros para el proyecto.

Networking y Presentación del Proyecto Final

Objetivos:
- Presentar los proyectos finales de los participantes de manera profesional.
- Recibir retroalimentación constructiva de compañeros, facilitadores y profesionales invitados.
- Establecer conexiones con otros participantes y profesionales del sector a través de actividades de networking.

Contenidos:

1. Preparación para la Presentación

Objetivo: Asegurar que los participantes estén listos para presentar sus proyectos de manera clara y efectiva.

1. **Revisión de la Presentación:**
   - **Estructura de la Presentación:**
     - Introducción: Breve resumen del proyecto y los objetivos.
     - Metodología: Descripción del enfoque y las técnicas utilizadas.
     - Resultados: Presentación de los hallazgos y resultados del proyecto.
     - Conclusiones: Reflexiones finales y posibles mejoras futuras.
   - **Materiales de Apoyo:**
     - Diapositivas con gráficos, tablas y puntos clave.
     - Código fuente y documentación del proyecto.
     - Gráficos y visualizaciones de datos relevantes.

2. **Práctica de la Presentación:**
   - Ensayo de la presentación para asegurar fluidez y claridad.
   - Preparación para responder posibles preguntas del público.
   - Asegurarse de que todo el equipo esté coordinado y listo para presentar.

2. Presentación del Proyecto Final

Objetivo: Permitir que cada equipo presente su proyecto, demostrando sus logros y el impacto de su trabajo.

1. **Configuración del Taller:**
   - **Moderador:** Designar a un moderador para introducir a cada equipo y gestionar el tiempo y las preguntas.
   - **Agenda:** Establecer un cronograma con tiempos asignados para cada presentación (por ejemplo, 10 minutos por equipo más 5 minutos para preguntas).

2. **Realización de Presentaciones:**
   - **Introducción por el Moderador:** Breve introducción sobre el propósito del taller y la agenda del día.
   - **Presentaciones de Equipos:** Cada equipo presenta su proyecto siguiendo la estructura preparada.
   - **Sesión de Preguntas y Retroalimentación:** Después de cada presentación, abrir el espacio para preguntas del público y proporcionar retroalimentación constructiva.

3. **Evaluación y Retroalimentación:**
   - **Criterios de Evaluación:** Evaluar cada presentación en términos de claridad, metodología, resultados y habilidades de presentación.
   - **Comentarios Constructivos:** Proporcionar retroalimentación específica y constructiva para ayudar a los participantes a mejorar.

3. Actividades de Networking

Objetivo: Facilitar el networking entre los estudiantes, profesionales de la industria y potenciales empleadores, creando oportunidades para establecer conexiones valiosas y explorar futuras colaboraciones o oportunidades de empleo.

1. **Actividades de Networking:**
   - **Speed Networking:** Organizar sesiones rápidas donde los estudiantes pueden interactuar con diferentes profesionales en intervalos cortos de tiempo.
   - **Mesas Redondas Temáticas:** Crear mesas redondas enfocadas en temas específicos relacionados con la inteligencia artificial, donde los estudiantes pueden discutir y compartir ideas con expertos.
   - **Intercambio de Contactos:** Proporcionar un espacio donde los estudiantes y los profesionales puedan intercambiar tarjetas de visita, perfiles de LinkedIn u otros medios de contacto.

2. **Consejos para el Networking:**
   - **Presentación Personal:** Enseñar a los estudiantes cómo presentarse de manera efectiva en un entorno de networking.
   - **Elevator Pitch:** Ayudar a los estudiantes a preparar un breve discurso que resuma quiénes son, qué hacen y qué buscan.
   - **Seguimiento:** Aconsejar a los estudiantes sobre la importancia de hacer un seguimiento con las personas que conozcan durante el evento.

Proyecto (lección 12)

colocar una indicación referente a subir el proyecto final y una imagen

English Code (lección 13) Markup languages HTML – HTML5

revisar documento drive estructura curso bootcmap, pero dejarlo para el final.

English Code (lección 14) Programming Language Python

revisar documento drive estructura curso bootcmap, pero dejarlo para el final.

Zona de Recarga (lección 15) Pensamiento Crítico en la Resolución de Problemas de IA

Pensamiento Crítico en la Resolución de Problemas de IA

Objetivos:
- Desarrollar la capacidad de pensar críticamente y de manera analítica.
- Aplicar habilidades de pensamiento crítico para identificar y resolver problemas en proyectos de IA.
- Fomentar una actitud proactiva hacia la identificación y resolución de problemas.

Contenidos:

1. Introducción al Pensamiento Crítico

Objetivo: Comprender los fundamentos del pensamiento crítico y su importancia en el campo de la inteligencia artificial.

1. **Definición de Pensamiento Crítico:**
   - Qué es el pensamiento crítico.
   - Importancia del pensamiento crítico en la resolución de problemas complejos.

2. **Componentes del Pensamiento Crítico:**
   - Análisis.
   - Evaluación.
   - Inferencia.
   - Explicación.
   - Autorregulación.

3. **Características de un Pensador Crítico:**
   - Curiosidad.
   - Escepticismo saludable.
   - Humildad intelectual.
   - Perseverancia.

2. Aplicación del Pensamiento Crítico en Proyectos de IA

Objetivo: Aplicar habilidades de pensamiento crítico en la identificación y resolución de problemas específicos en proyectos de inteligencia artificial.

1. **Identificación de Problemas:**
   - Cómo identificar problemas en datos y modelos de IA.

- Análisis de datos para detectar anomalías y patrones.
```python
import pandas as pd
df = pd.read_csv('data.csv')
print(df.describe())
```

2. **Evaluación de Soluciones:**
   - Evaluar diferentes enfoques para resolver un problema.
   - Comparar ventajas y desventajas de cada enfoque.
   ```python
   from sklearn.model_selection import train_test_split
   from sklearn.linear_model import LogisticRegression
   from sklearn.tree import DecisionTreeClassifier

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
   log_reg = LogisticRegression().fit(X_train, y_train)
   tree_clf = DecisionTreeClassifier().fit(X_train, y_train)

   log_reg_accuracy = log_reg.score(X_test, y_test)
   tree_clf_accuracy = tree_clf.score(X_test, y_test)

   print(f'Logistic Regression Accuracy: {log_reg_accuracy}')
   print(f'Decision Tree Accuracy: {tree_clf_accuracy}')
   ```

3. **Solución de Problemas:**
   - Desarrollo de estrategias para resolver problemas identificados.
   - Implementación de soluciones efectivas.
   ```python
   # Solución de problema de datos faltantes
   df.fillna(df.mean(), inplace=True)
   ```

3. Estudios de Caso y Ejercicios Prácticos

Objetivo: Poner en práctica las habilidades de pensamiento crítico mediante estudios de caso y ejercicios prácticos.

1. **Estudio de Caso 1:**
   - Descripción de un problema de IA real (por ejemplo, detección de fraudes).
   - Análisis del problema y discusión de posibles soluciones.
   ```python
   # Análisis de transacciones bancarias para detección de fraudes
   df['fraud'] = (df['transaction_amount'] > 1000).astype(int)
   print(df.groupby('fraud').size())
   ```

2. **Estudio de Caso 2:**
   - Descripción de un problema de modelado predictivo (por ejemplo, predicción de ventas).
   - Evaluación de diferentes modelos y selección del mejor enfoque.
   ```python
   # Predicción de ventas utilizando regresión lineal
   from sklearn.linear_model import LinearRegression
   model = LinearRegression().fit(X_train, y_train)
   predictions = model.predict(X_test)
   print(predictions)
   ```

3. **Ejercicios Prácticos:**
   - Resolución de problemas específicos utilizando pensamiento crítico.
   - Discusión en grupo de las soluciones propuestas.

Zona de Recarga (lección 16) Importancia de la Colaboración en Proyectos de IA

Objetivo: Comprender la importancia de la colaboración y el trabajo en equipo en el desarrollo de proyectos de inteligencia artificial.

1. **Beneficios de la Colaboración:**
   - Diversidad de perspectivas y habilidades.
   - Mejora en la creatividad y la innovación.
   - Aumento de la eficiencia y la productividad.

2. **Desafíos de la Colaboración:**
   - Coordinación de tareas y responsabilidades.
   - Gestión de conflictos y diferencias de opinión.
   - Comunicación efectiva en equipos multidisciplinarios.

2. Técnicas de Colaboración y Trabajo en Equipo

Objetivo: Aprender y aplicar técnicas efectivas de colaboración y trabajo en equipo en proyectos de IA.

1. **Metodologías de Trabajo en Equipo:**
   - Metodología ágil (Scrum, Kanban).
   - Planificación de sprints y retrospectivas.
   ```python
   # Ejemplo de planificación de sprint con tareas asignadas
   sprint_plan = {
       'Data Collection': ['Alice', 'Bob'],
       'Model Development': ['Charlie', 'Dave'],
       'Evaluation and Testing': ['Eve', 'Frank']
   }
   ```

```
print(sprint_plan)
```

2. **Herramientas de Colaboración:**
   - Uso de herramientas como GitHub para la gestión de versiones y colaboración en código.
   - Plataformas de comunicación y gestión de proyectos (Slack, Trello).
   ```python
   # Ejemplo de uso de GitHub para colaboración
   !git clone https://github.com/usuario/proyecto-ia.git
   !cd proyecto-ia && git checkout -b nueva-rama
   !git add .
   !git commit -m "Añadir nuevo modelo"
   !git push origin nueva-rama
   ```

3. **Técnicas de Resolución de Problemas en Equipo:**
   - Brainstorming y lluvia de ideas.
   - Análisis de causa raíz (Diagrama de Ishikawa).
   - Técnicas de toma de decisiones (Matriz de Decisión).
   ```python
   # Ejemplo de análisis de causa raíz utilizando pandas
   df['root_cause'] = df['issue'].apply(lambda x: 'Data Issue' if 'data' in x else 'Model Issue')
   print(df.groupby('root_cause').size())
   ```

3. Ejercicios Prácticos y Dinámicas de Grupo

Objetivo: Poner en práctica las habilidades de colaboración y trabajo en equipo mediante ejercicios prácticos y dinámicas de grupo.

1. **Ejercicio Práctico 1:**
   - Trabajar en equipo para resolver un problema de IA específico (por ejemplo, mejorar la precisión de un modelo).
   - Discusión de estrategias y soluciones propuestas.
   ```python
   # Mejora de la precisión del modelo mediante la optimización de hiperparámetros
   from sklearn.model_selection import GridSearchCV
   param_grid = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01]}
   grid = GridSearchCV(SVC(), param_grid, refit=True)
   grid.fit(X_train, y_train)
   print(grid.best_params_)
   ```

2. **Dinámica de Grupo:**
   - Simulación de una reunión de equipo para planificar y coordinar un proyecto de IA.
   - Resolución de conflictos y toma de decisiones colaborativa.
   ```python
   ```

```
# Ejemplo de planificación de tareas utilizando un tablero Kanban
kanban_board = {
    'To Do': ['Recolección de datos', 'Preprocesamiento'],
    'In Progress': ['Entrenamiento del modelo'],
    'Done': ['Exploración de datos']
}
print(kanban_board)
```

Misión 2: Aplicaciones Prácticas de la IA

Mapa de Ruta - Misión 2: Aplicaciones Prácticas de la IA

Mapa de Ruta

Objetivo General: Proporcionar a los participantes una comprensión sólida de las aplicaciones prácticas de la inteligencia artificial, con un enfoque en el desarrollo de chatbots y la implementación de técnicas de machine learning en problemas prácticos. Los estudiantes aprenderán a aplicar conceptos de IA en casos de uso reales y desarrollarán un proyecto práctico que integre estas técnicas.

Objetivos del Bootcamp

1. **Aplicaciones Prácticas de IA:**
   - Entender las diferentes aplicaciones de la inteligencia artificial en diversos dominios.
   - Conocer ejemplos reales de cómo se utiliza la IA para resolver problemas prácticos.

2. **Desarrollo de Chatbots:**
   - Aprender a diseñar y desarrollar chatbots utilizando técnicas de IA.
   - Implementar chatbots para casos de uso específicos.

3. **Técnicas de Machine Learning:**
   - Aplicar técnicas de machine learning a problemas prácticos.
   - Desarrollar y evaluar modelos de machine learning en casos de uso reales.

Habilidades Digitales a Desarrollar

1. **Implementación de Técnicas de IA:**
   - Capacidad para aplicar técnicas de machine learning a problemas prácticos.
   - Habilidad para diseñar y desarrollar chatbots utilizando técnicas de IA.

2. **Desarrollo de Proyectos Prácticos:**
   - Experiencia en el desarrollo de proyectos prácticos de IA.
   - Capacidad para integrar técnicas de IA en aplicaciones reales.

3. **Evaluación de Modelos:**
   - Habilidad para evaluar y ajustar modelos de machine learning.
   - Capacidad para interpretar y comunicar los resultados de los modelos de IA.

Contenidos a Abordar

1. **Aplicaciones de IA:**
   - Casos de uso de la IA en diferentes dominios (salud, finanzas, transporte, marketing, etc.).
   - Ejemplos prácticos de cómo se utiliza la IA para resolver problemas específicos.

2. **Desarrollo de Chatbots:**
   - Diseño y arquitectura de chatbots.
   - Implementación de chatbots utilizando bibliotecas y frameworks de IA (por ejemplo, NLTK, Rasa, TensorFlow).

3. **Técnicas de Machine Learning:**
   - Aplicación de técnicas de machine learning a problemas prácticos.
   - Implementación y evaluación de modelos de machine learning.

Propósitos Educativos del Bootcamp

- **Comprensión Práctica:** Proporcionar a los participantes una comprensión práctica de cómo se utilizan las técnicas de IA para resolver problemas reales.
- **Desarrollo de Habilidades:** Desarrollar habilidades en el diseño, implementación y evaluación de chatbots y modelos de machine learning.
- **Aplicación de Conocimientos:** Aplicar los conocimientos adquiridos en proyectos prácticos que aborden problemas específicos en diversos dominios.

Alineación con el Mercado Laboral

El contenido del bootcamp está alineado con las necesidades del mercado laboral en el ámbito de la inteligencia artificial. Los estudiantes adquirirán competencias demandadas por la industria, incluyendo:

- Habilidades en el desarrollo de chatbots y aplicaciones de IA.
- Experiencia práctica en la implementación y evaluación de modelos de machine learning.
- Capacidad para aplicar técnicas de IA en casos de uso reales.

Preparación para Desafíos Digitales

Los participantes estarán preparados para enfrentar desafíos digitales relacionados con la inteligencia artificial, incluyendo:

- Desarrollo de chatbots y asistentes virtuales.
- Implementación de técnicas de machine learning en problemas prácticos.
- Evaluación y ajuste de modelos de IA.

Contenidos de la Misión 2

Entrenamiento

- **Mapa de Ruta:** Aplicaciones de IA en diferentes dominios, desarrollo de chatbots.
  - **Objetivo:** Proporcionar una comprensión práctica de las aplicaciones de IA y el desarrollo de chatbots.
  - **Habilidades Digitales:** Implementación de técnicas de IA, desarrollo de chatbots, evaluación de modelos.
  - **Contenidos:**
    - Aplicaciones de IA en salud, finanzas, transporte, marketing, etc.
    - Diseño y arquitectura de chatbots.
    - Implementación de chatbots utilizando bibliotecas y frameworks de IA.

- **Preparación (15 horas):** Técnicas de machine learning aplicadas a problemas prácticos.
  - Aplicaciones de IA en la industria y la vida cotidiana.
  - Introducción a los chatbots y asistentes virtuales.
  - Implementación de un chatbot básico.
  - Ajuste y mejora del chatbot.

- **English Code (8 horas):** Terminología en inglés sobre aplicaciones de IA.
  - **Contenido:**
    - Vocabulario técnico en aplicaciones de IA.
    - Lectura y comprensión de documentación técnica.
    - Redacción de comentarios y documentación en inglés.

Experiencia

- **Simulación (10 horas):** Desarrollo de un chatbot utilizando técnicas de IA.
  - Configuración del entorno para el desarrollo de chatbots.
  - Implementación de un chatbot utilizando bibliotecas de Python.
  - Entrenamiento y prueba del chatbot.
  - Evaluación del desempeño del chatbot.

- **Cápsulas:** Recursos multimedia sobre aplicaciones prácticas de IA.
  - **Contenido:** Videos y tutoriales que refuercen los conceptos aprendidos y muestren ejemplos prácticos.

- **Zona de Recarga (5 horas):** Resolución de problemas y pensamiento crítico en aplicaciones de IA.
  - **Estrategias de resolución de problemas en proyectos de IA.
  - **Innovación y creatividad en el desarrollo de aplicaciones de IA.

Conexión

- **Prosumidores (3 horas):** Implementación de un proyecto práctico de IA.

  - **Contenido:** Desarrollo de un proyecto práctico que aplique técnicas de IA en un contexto real.

- **Co-creación (10 horas):** Desarrollo de una aplicación colaborativa de IA para un sector específico.
  - Definición del proyecto colaborativo.
  - Desarrollo y ajustes del proyecto.
  - Presentación del proyecto y retroalimentación.

- **Satélites (2 horas):** Talleres de presentación de proyectos y networking.
  - **Sesión única (2 horas):** Taller de presentación de proyectos y actividades de networking.

Preparación (lección 1) Aplicaciones de IA en la Industria y la Vida Cotidiana

Objetivos:
- Entender cómo se aplican las técnicas de IA en diversos dominios industriales y cotidianos.
- Explorar ejemplos prácticos de la utilización de IA para resolver problemas reales.

Contenidos:

1. Introducción a las Aplicaciones de IA

Objetivo: Proporcionar una visión general de las aplicaciones de la inteligencia artificial en diferentes sectores.

1. **Dominios de Aplicación de la IA:**
   - Salud.
   - Finanzas.
   - Transporte.
   - Marketing.
   - Educación.
   - Agricultura.

2. **Impacto de la IA en la Vida Cotidiana:**
   - Asistentes virtuales (Siri, Alexa).
   - Sistemas de recomendación (Netflix, Amazon).
   - Vehículos autónomos.
   - Detección de fraudes.

2. Aplicaciones de IA en el Sector Salud

Objetivo: Explorar cómo se utiliza la IA en el sector salud para mejorar diagnósticos, tratamientos y gestión de datos.

1. **Diagnóstico Asistido por IA:**
   - Análisis de imágenes médicas (rayos X, resonancias magnéticas).
   - Detección temprana de enfermedades (cáncer, enfermedades cardíacas).

2. **Predicción de Resultados de Tratamientos:**
   - Modelos predictivos para la personalización de tratamientos.
   - Optimización de recursos hospitalarios.

3. **Gestión de Datos en Salud:**
   - Análisis de grandes volúmenes de datos de pacientes.
   - Sistemas de gestión de registros médicos electrónicos.

3. Aplicaciones de IA en el Sector Financiero

Objetivo: Entender cómo se aplica la IA en el sector financiero para mejorar la seguridad, eficiencia y toma de decisiones.

1. **Detección de Fraudes:**
   - Análisis de transacciones para identificar patrones sospechosos.
   - Implementación de sistemas de alerta temprana.

2. **Modelos de Crédito y Riesgo:**
   - Evaluación del riesgo crediticio utilizando machine learning.
   - Predicción de incumplimiento de pagos.

3. **Trading Algorítmico:**
   - Uso de algoritmos para optimizar operaciones bursátiles.
   - Predicción de tendencias del mercado financiero.

4. Aplicaciones de IA en el Marketing y las Ventas

Objetivo: Explorar cómo la IA se utiliza en marketing y ventas para mejorar la segmentación, personalización y análisis del comportamiento del consumidor.

1. **Sistemas de Recomendación:**
   - Personalización de recomendaciones de productos y servicios.
   ```python
   from sklearn.neighbors import NearestNeighbors
   import numpy as np

   user_ratings = np.array([[5, 4, 0, 1], [3, 2, 0, 5], [0, 0, 5, 4]])
   model = NearestNeighbors(n_neighbors=2, algorithm='auto').fit(user_ratings)
   distances, indices = model.kneighbors(user_ratings)
   print(indices)
   ```

2. **Análisis de Sentimientos:**
   - Análisis de comentarios y opiniones en redes sociales.
   ```python
   from textblob import TextBlob

   comments = ["I love this product!", "This is the worst service ever."]
   sentiments = [TextBlob(comment).sentiment.polarity for comment in comments]
   print(sentiments)
   ```

3. **Segmentación de Clientes:**
   - Agrupamiento de clientes basado en su comportamiento de compra.
   ```python
   from sklearn.cluster import KMeans

   customer_data = np.array([[23, 50000], [35, 60000], [45, 80000], [50, 100000]])
   kmeans = KMeans(n_clusters=2, random_state=0).fit(customer_data)
   print(kmeans.labels_)
   ```

Preparación (lección 2) Introducción a los Chatbots y Asistentes Virtuales

Objetivos:
- Comprender los fundamentos del desarrollo de chatbots y asistentes virtuales.
- Aprender a diseñar la arquitectura de un chatbot.
- Implementar un chatbot básico utilizando bibliotecas de IA.

Contenidos:

1. Fundamentos de los Chatbots

Objetivo: Proporcionar una comprensión básica de los chatbots, su arquitectura y casos de uso.

1. **Qué es un Chatbot:**
   - Definición y características.
   - Tipos de chatbots: Basados en reglas y basados en IA.

2. **Arquitectura de un Chatbot:**
   - Componentes principales: Entrada del usuario, procesamiento del lenguaje natural (NLP), generación de respuestas.
   - Flujo de trabajo de un chatbot.

3. **Casos de Uso de los Chatbots:**
   - Atención al cliente.

- Asistencia en compras en línea.
- Automatización de tareas administrativas.

2. Introducción al Procesamiento del Lenguaje Natural (NLP)

Objetivo: Entender los conceptos básicos del NLP y cómo se aplican en el desarrollo de chatbots.

1. **Fundamentos del NLP:**
   - Tokenización.
   - Lemmatización y stemming.
   - Análisis sintáctico y semántico.

2. **Bibliotecas de NLP en Python:**
   - NLTK.
   - SpaCy.
   ```python
   import spacy

   nlp = spacy.load("en_core_web_sm")
   doc = nlp("Chatbots are revolutionizing customer service.")
   for token in doc:
       print(token.text, token.lemma_, token.pos_)
   ```

3. Diseño y Desarrollo de un Chatbot Básico

Objetivo: Implementar un chatbot básico utilizando bibliotecas de Python.

1. **Configuración del Entorno de Desarrollo:**
   - Instalación de bibliotecas necesarias.
   ```bash
   pip install rasa
   ```

2. **Desarrollo de un Chatbot con Rasa:**
   - Creación de un chatbot básico para responder preguntas frecuentes.
   - Configuración del dominio y las historias del chatbot.
   ```yaml
   # domain.yml
   intents:
     - greet
     - ask_weather

   responses:
     utter_greet:
       - text: "Hello! How can I help you today?"
     utter_ask_weather:
   ```

```
      - text: "The weather is sunny."

  stories:
    - story: greet user
      steps:
      - intent: greet
      - action: utter_greet
```

3. **Entrenamiento y Prueba del Chatbot:**
   - Entrenamiento del modelo de NLP.
   - Prueba del chatbot en un entorno simulado.
   ```bash
   rasa train
   rasa shell
   ```

Preparación (lección 3) Implementación de un Chatbot Básico

Objetivos:
- Implementar un chatbot básico utilizando técnicas de NLP.
- Entrenar y probar el chatbot para asegurar su funcionalidad.
- Mejorar el chatbot mediante el ajuste de parámetros y la adición de nuevas funcionalidades.

Contenidos:

1. Implementación del Chatbot

Objetivo: Desarrollar e implementar un chatbot básico que responda preguntas frecuentes.

1. **Desarrollo del Modelo de NLP:**
   - Entrenamiento del modelo utilizando ejemplos de diálogo.
   ```python
   from rasa.nlu.training_data import load_data
   from rasa.nlu.model import Trainer
   from rasa.nlu import config

   training_data = load_data("data/nlu.md")
   trainer = Trainer(config.load("config.yml"))
   interpreter = trainer.train(training_data)
   ```

2. **Configuración de Respuestas del Chatbot:**
   - Definición de respuestas automáticas a las intenciones del usuario.
   ```yaml
   ```

```yaml
# domain.yml
responses:
  utter_greet:
    - text: "Hello! How can I assist you today?"
  utter_ask_weather:
    - text: "It's sunny outside."
```

3. **Prueba del Chatbot:**
   - Evaluación de la precisión del chatbot mediante pruebas interactivas.
   ```bash
   rasa shell
   ```

2. Mejora y Ajuste del Chatbot

Objetivo: Mejorar el chatbot mediante el ajuste de parámetros y la adición de nuevas funcionalidades.

1. **Ajuste de Parámetros del Modelo:**
   - Optimización del modelo de NLP para mejorar la precisión de las respuestas.
   ```yaml
   # config.yml
   pipeline:
     - name: WhitespaceTokenizer
     - name: CountVectorsFeaturizer
     - name: DIETClassifier
       epochs: 100
   ```

2. **Adición de Nuevas Funcionalidades:**
   - Implementación de nuevas intenciones y respuestas.
   ```yaml
   # domain.yml
   intents:
     - greet
     - ask_weather
     - ask_time

   responses:
     utter_ask_time:
       - text: "The current time is 3 PM."
   ```

3. **Evaluación y Validación del Chatbot:**
   - Pruebas finales para asegurar la funcionalidad y precisión del chatbot.
   ```bash
   ```

```
rasa test
```

Preparación y Simulación (lección 4) Ajuste, Mejora del Chatbot y Configuración del Entorno para el Desarrollo de Chatbots

Ajuste y Mejora del Chatbot

Objetivos:
- Evaluar el rendimiento del chatbot y realizar ajustes para mejorar su precisión.
- Documentar el desarrollo y las mejoras realizadas en el chatbot.

Contenidos:

1. Evaluación del Desempeño del Chatbot

Objetivo: Evaluar el rendimiento del chatbot mediante métricas de precisión y retroalimentación del usuario.

1. **Métricas de Evaluación:**
   - Precisión, recall, F1-score.
   ```python
   from sklearn.metrics import classification_report

   y_true = ["greet", "ask_weather"]
   y_pred = ["greet", "ask_weather"]
   print(classification_report(y_true, y_pred))
   ```

2. **Recopilación de Retroalimentación:**
   - Obtener comentarios de los usuarios para identificar áreas de mejora.

2. Ajuste y Optimización del Chatbot

Objetivo: Realizar ajustes en el modelo y la configuración del chatbot para mejorar su rendimiento.

1. **Optimización del Modelo:**
   - Ajuste de hiperparámetros y reentrenamiento del modelo.
   ```yaml
   # config.yml
   pipeline:
     - name: WhitespaceTokenizer
     - name: CountVectorsFeaturizer
     - name: DIETClassifier
       epochs: 150
```

```

2. **Implementación de Mejoras Basadas en Retroalimentación:**
   - Incorporación de nuevas intenciones y respuestas basadas en la retroalimentación recibida.
   ```yaml
   # domain.yml
   intents:
     - greet
     - ask_weather
     - ask_time
     - ask_date

   responses:
     utter_ask_date:
       - text: "Today's date is April 1st."
   ```

3. Documentación y Presentación del Chatbot

Objetivo: Documentar el desarrollo del chatbot y preparar una presentación para compartir los resultados.

1. **Documentación del Proceso:**
   - Redacción de un informe detallado sobre el desarrollo, entrenamiento y ajuste del chatbot.
   ```markdown
   # Desarrollo de un Chatbot Básico

   ## Introducción
   Descripción del proyecto y objetivos.

   ## Implementación
   Detalles sobre la configuración del entorno, el modelo de NLP y las respuestas.

   ## Evaluación
   Métricas de evaluación y retroalimentación del usuario.

   ## Mejoras
   Ajustes y optimizaciones realizadas para mejorar el rendimiento del chatbot.
   ```

2. **Preparación de la Presentación:**
   - Creación de diapositivas para presentar el desarrollo y los resultados del chatbot.
   - Incluir gráficos y tablas para ilustrar los resultados obtenidos.

Configuración del Entorno para el Desarrollo de Chatbots
   ```

Objetivos:
- Configurar el entorno de desarrollo necesario para implementar un chatbot.
- Instalar y verificar las bibliotecas y herramientas requeridas.

Contenidos:

1. Instalación de Herramientas y Bibliotecas

Objetivo: Asegurar que todos los participantes tengan el entorno de desarrollo correctamente configurado.

1. **Instalación de Python y Bibliotecas Esenciales:**
   - Instalación de Python (si no está ya instalado).
   - Instalación de Rasa y otras bibliotecas necesarias.
   ```bash
   pip install rasa
   pip install rasa[transformers]
   pip install spacy
   python -m spacy download en_core_web_sm
   ```

2. **Configuración del Entorno de Desarrollo:**
   - Uso de Visual Studio Code o Jupyter Notebook para el desarrollo.
   - Instalación de extensiones relevantes para la codificación en Python.

2. Verificación de la Configuración

Objetivo: Verificar que todas las instalaciones y configuraciones estén funcionando correctamente.

1. **Prueba de Instalación de Rasa:**
   - Creación de un nuevo proyecto de Rasa.
   ```bash
   rasa init
   ```
   - Ejecución del servidor de Rasa para verificar la instalación.
   ```bash
   rasa shell
   ```

2. **Verificación de SpaCy:**
   - Ejecución de un script de prueba para asegurarse de que SpaCy está funcionando correctamente.
   ```python
   import spacy
   nlp = spacy.load("en_core_web_sm")
   doc = nlp("Chatbots are transforming customer service.")
   ```

```
for token in doc:
    print(token.text, token.lemma_, token.pos_)
```

Simulación (lección 5) Implementación del Chatbot

1. Creación del Proyecto de Chatbot

Objetivo: Configurar el proyecto de Rasa y definir las intenciones y entidades.

1. **Estructura del Proyecto:**
   - Creación de la estructura básica del proyecto de Rasa.
   ```bash
   rasa init --no-prompt
   ```

2. **Definición de Intenciones y Entidades:**
   - Añadir nuevas intenciones y entidades en el archivo de datos de entrenamiento (`nlu.yml`).
   ```yaml
   # nlu.yml
   nlu:
   - intent: greet
     examples: |
       - hello
       - hi
       - hey
       - good morning
       - good evening

   - intent: ask_weather
     examples: |
       - what's the weather like today?
       - tell me the weather
       - will it rain today?
   ```

3. **Configuración del Dominio:**
   - Definir las respuestas del chatbot en el archivo de dominio (`domain.yml`).
   ```yaml
   # domain.yml
   intents:
     - greet
     - ask_weather

   responses:
     utter_greet:
```

```
    - text: "Hello! How can I assist you today?"
  utter_ask_weather:
    - text: "The weather is sunny."
```

2. Entrenamiento del Modelo de NLP

Objetivo: Entrenar el modelo de Rasa para entender las intenciones del usuario.

1. **Entrenamiento del Modelo:**
   - Ejecutar el comando de entrenamiento de Rasa.
   ```bash
   rasa train
   ```

2. **Evaluación del Modelo:**
   - Verificación de la precisión del modelo utilizando datos de prueba.
   ```bash
   rasa test nlu
   ```

3. Desarrollo de Acciones Personalizadas

Objetivo: Implementar acciones personalizadas para mejorar la funcionalidad del chatbot.

1. **Definición de Acciones Personalizadas:**
   - Crear un archivo de acciones (`actions.py`) y definir acciones personalizadas.
   ```python
   from rasa_sdk import Action, Tracker
   from rasa_sdk.executor import CollectingDispatcher

   class ActionTellWeather(Action):
       def name(self) -> str:
           return "action_tell_weather"

       def run(self, dispatcher: CollectingDispatcher,
               tracker: Tracker,
               domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
           dispatcher.utter_message(text="The weather is sunny.")
           return []
   ```

2. **Configuración de Acciones Personalizadas:**
   - Añadir las acciones personalizadas en el archivo de dominio (`domain.yml`).
   ```yaml
   actions:
     - action_tell_weather
```

```

3. **Ejecutar el Servidor de Acciones:**
   - Iniciar el servidor de acciones para que el chatbot pueda utilizar las acciones personalizadas.
   ```bash
   rasa run actions
   ```

4. Prueba y Depuración del Chatbot

Objetivo: Probar el chatbot en un entorno interactivo y depurar cualquier problema.

1. **Prueba del Chatbot:**
   - Ejecutar el chatbot en modo interactivo para realizar pruebas.
   ```bash
   rasa shell
   ```

2. **Depuración de Problemas:**
   - Identificar y solucionar problemas encontrados durante la prueba.
   - Ajustar el modelo y las respuestas según sea necesario.

Simulación (lección 6) Entrenamiento y Prueba del Chatbot

1. Entrenamiento del Chatbot con Datos Reales

Objetivo: Entrenar el chatbot utilizando un conjunto de datos más amplio y realista.

1. **Recolección de Datos:**
   - Obtener un conjunto de datos de diálogo para entrenar el chatbot.
   - Formatear los datos según las necesidades del proyecto de Rasa.

2. **Entrenamiento del Modelo con Datos Reales:**
   - Añadir nuevos ejemplos de intenciones y entidades en `nlu.yml`.
   - Entrenar el modelo con los nuevos datos.
   ```bash
   rasa train
   ```

3. **Evaluación del Modelo:**
   - Evaluar la precisión del modelo utilizando un conjunto de datos de prueba.
   ```bash
   rasa test nlu
   ```

2. Prueba del Chatbot en un Entorno Simulado
   ```

Objetivo: Evaluar el rendimiento del chatbot en un entorno controlado.

1. **Configuración del Entorno de Prueba:**
   - Crear un entorno simulado para probar las respuestas del chatbot.
   - Utilizar Rasa X para una interfaz gráfica de pruebas.
   ```bash
   rasa x
   ```

2. **Prueba Interactiva:**
   - Interactuar con el chatbot y registrar las respuestas.
   - Identificar áreas de mejora.

3. Mejora de la Precisión del Chatbot

Objetivo: Optimizar el rendimiento del chatbot mediante ajustes y mejoras.

1. **Ajuste de Hiperparámetros:**
   - Ajustar los hiperparámetros del modelo de NLP para mejorar la precisión.
   ```yaml
   pipeline:
     - name: WhitespaceTokenizer
     - name: CountVectorsFeaturizer
     - name: DIETClassifier
       epochs: 200
   ```

2. **Incorporación de Nuevos Datos:**
   - Añadir nuevos ejemplos y mejorar la diversidad del conjunto de datos.
   - Reentrenar el modelo con los datos adicionales.
   ```bash
   rasa train
   ```

3. **Evaluación Continua:**
   - Realizar pruebas periódicas para asegurar que el chatbot mantiene su precisión y funcionalidad.
   ```bash
   rasa test nlu
   ```

Simulación y prosumidores (lección 7) Evaluación del Desempeño del Chatbot e Implementación de un Proyecto Práctico de IA

1. Evaluación del Desempeño Final

Objetivo: Evaluar el chatbot utilizando métricas de precisión y retroalimentación de los usuarios.

1. **Métricas de Evaluación:**
   - Precisión, recall, F1-score.
   ```python
   from sklearn.metrics import classification_report

   y_true = ["greet", "ask_weather"]
   y_pred = ["greet", "ask_weather"]
   print(classification_report(y_true, y_pred))
   ```

2. **Recopilación de Retroalimentación:**
   - Obtener comentarios de los usuarios para identificar áreas de mejora.

2. Documentación y Presentación

Objetivo: Documentar el desarrollo y los resultados del chatbot, y preparar una presentación final.

1. **Documentación del Proceso:**
   - Redacción de un informe detallado sobre el desarrollo, entrenamiento y evaluación del chatbot.
   ```markdown
   # Desarrollo de un Chatbot Básico

   ## Introducción
   Descripción del proyecto y objetivos.

   ## Implementación
   Detalles sobre la configuración del entorno, el modelo de NLP y las respuestas.

   ## Evaluación
   Métricas de evaluación y retroalimentación del usuario.

   ## Mejoras
   Ajustes y optimizaciones realizadas para mejorar el rendimiento del chatbot.
   ```

2. **Preparación de la Presentación:**
   - Creación de diapositivas para presentar el desarrollo y los resultados del chatbot.
   - Incluir gráficos y tablas para ilustrar los resultados obtenidos.

Implementación de un Proyecto Práctico de IA

Objetivos:
- Identificar un problema específico que pueda ser resuelto utilizando técnicas de inteligencia artificial.
- Desarrollar un modelo de IA para abordar el problema seleccionado.
- Evaluar y presentar los resultados del proyecto.

Contenidos:

1. Selección y Definición del Proyecto

Objetivo: Identificar un problema específico y definir el alcance del proyecto de IA.

1. **Brainstorming de Ideas:**
   - **Ejemplos de Proyectos Simples:**
     - Clasificación de correos electrónicos (spam vs. no spam).
     - Predicción de precios de viviendas.
     - Análisis de sentimientos en comentarios de redes sociales.
     - Sistema de recomendación simple (por ejemplo, recomendación de películas).

2. **Definición del Proyecto:**
   - **Título del Proyecto:** Clasificación de correos electrónicos como spam o no spam.
   - **Descripción del Problema:** Desarrollar un modelo de IA que clasifique correos electrónicos en dos categorías: spam y no spam.
   - **Objetivos del Proyecto:**
     - Recolectar y preparar un conjunto de datos de correos electrónicos.
     - Implementar un algoritmo de clasificación (por ejemplo, Naive Bayes).
     - Evaluar el rendimiento del modelo.

2. Desarrollo del Proyecto

Objetivo: Implementar el modelo de IA y realizar el análisis correspondiente.

1. **Recolección y Preparación de Datos:**
   - **Carga del Conjunto de Datos:**
     - Uso de un conjunto de datos de correos electrónicos (por ejemplo, Enron Email Dataset).
     ```python
     import pandas as pd

     # Cargar datos (ejemplo de dataset de correos electrónicos)
     data = pd.read_csv('emails.csv')
     print(data.head())
     ```

   - **Preprocesamiento de Datos:**
     - Limpieza de texto (eliminación de puntuación, conversión a minúsculas, eliminación de palabras irrelevantes).

```python
import re
from nltk.corpus import stopwords

def preprocess_text(text):
    text = re.sub(r'\W', ' ', text)
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    return text

data['text'] = data['text'].apply(preprocess_text)
stop_words = set(stopwords.words('english'))
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))
print(data.head())
```

2. **Implementación del Modelo de Clasificación:**
   - **Vectorización de Texto:**
     - Conversión del texto en vectores numéricos utilizando TF-IDF.
     ```python
     from sklearn.feature_extraction.text import TfidfVectorizer

     vectorizer = TfidfVectorizer()
     X = vectorizer.fit_transform(data['text'])
     y = data['spam']
     ```

   - **Entrenamiento del Modelo:**
     - División de los datos en conjuntos de entrenamiento y prueba.
     - Entrenamiento de un modelo de Naive Bayes.
     ```python
     from sklearn.model_selection import train_test_split
     from sklearn.naive_bayes import MultinomialNB

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
     model = MultinomialNB()
     model.fit(X_train, y_train)
     ```

   - **Evaluación del Modelo:**
     - Predicción en el conjunto de prueba y evaluación del rendimiento.
     ```python
     y_pred = model.predict(X_test)
     from sklearn.metrics import classification_report, accuracy_score
     print(classification_report(y_test, y_pred))
     print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
     ```

3. **Ajuste del Modelo:**
   - **Ajuste de Parámetros:**
     - Exploración de diferentes parámetros para mejorar el rendimiento del modelo.
     ```python
     from sklearn.model_selection import GridSearchCV

     param_grid = {'alpha': [0.1, 0.5, 1.0]}
     grid_search = GridSearchCV(MultinomialNB(), param_grid, cv=5)
     grid_search.fit(X_train, y_train)
     print(f'Best parameters: {grid_search.best_params_}')
     ```

4. **Visualización y Documentación:**
   - **Creación de Gráficos:**
     - Visualización de los resultados del modelo.
     ```python
     import matplotlib.pyplot as plt
     from sklearn.metrics import plot_confusion_matrix

     plot_confusion_matrix(model, X_test, y_test)
     plt.show()
     ```

   - **Documentación del Proceso:**
     - Redacción de un informe detallado sobre el proyecto, incluyendo la recolección de datos, preprocesamiento, implementación del modelo y evaluación.

3. Presentación de Resultados

Objetivo: Presentar el proyecto final, los resultados obtenidos y las conclusiones.

1. **Preparación de la Presentación:**
   - **Estructura de la Presentación:**
     - Introducción al proyecto y definición del problema.
     - Descripción del conjunto de datos y el preprocesamiento realizado.
     - Implementación del modelo y resultados obtenidos.
     - Conclusiones y posibles mejoras futuras.

   - **Materiales de Apoyo:**
     - Diapositivas con gráficos y tablas.
     - Código fuente documentado.

2. **Presentación del Proyecto:**
   - **Exposición del Proyecto:**
     - Presentación del problema, metodología y resultados.
     - Discusión de los desafíos enfrentados y cómo se resolvieron.

- **Sesión de Preguntas y Respuestas:**
  - Abrir espacio para preguntas y retroalimentación de los compañeros y facilitadores.

Cápsulas (lección 8)

- Videos y tutoriales sobre aplicaciones prácticas de IA.

Links de recursos según competencia especificas curados (yudi)

Co-creación (lección 9) Definición del Proyecto Colaborativo

Objetivos:
- Seleccionar un sector específico y definir el problema a resolver.
- Planificar el proyecto colaborativo, asignar roles y responsabilidades.
- Establecer metas y cronograma para el desarrollo del proyecto.

Contenidos:

1. Selección del Sector Específico y Problema a Resolver

Objetivo: Identificar un sector específico y definir un problema práctico que pueda ser resuelto utilizando técnicas de IA.

1. **Brainstorming de Sectores y Problemas:**
   - Educación: Predicción de rendimiento estudiantil.
   - Salud: Diagnóstico de enfermedades a partir de imágenes médicas.
   - Finanzas: Detección de fraudes en transacciones bancarias.
   - Marketing: Análisis de sentimientos en redes sociales.
   - Agricultura: Predicción de rendimientos de cultivos.

2. **Definición del Problema:**
   - **Sector Seleccionado:** Salud.
   - **Problema a Resolver:** Diagnóstico de enfermedades pulmonares a partir de imágenes de rayos X.
   - **Objetivo del Proyecto:** Desarrollar un modelo de IA que clasifique imágenes de rayos X en categorías de enfermedades pulmonares.

2. Planificación del Proyecto

Objetivo: Planificar el proyecto colaborativo y asignar roles y responsabilidades a los miembros del equipo.

1. **Definición del Alcance:**
   - Especificar qué se incluirá y qué no se incluirá en el proyecto.
   - Establecer criterios de éxito para el proyecto.

2. **Asignación de Roles y Responsabilidades:**

- Project Manager: Coordinación y supervisión general.
- Data Scientists: Recolección y preprocesamiento de datos.
- Machine Learning Engineers: Desarrollo e implementación del modelo.
- Data Analysts: Evaluación y análisis de resultados.

3. **Desarrollo de un Cronograma:**
   - Dividir el proyecto en fases y establecer plazos para cada fase.
   - Crear un calendario de reuniones y revisiones de progreso.

3. Establecimiento de Metas y Resultados Esperados

Objetivo: Definir metas a corto, mediano y largo plazo para el proyecto, y establecer resultados esperados.

1. **Metas a Corto Plazo:**
   - Recolección y preprocesamiento de datos de imágenes de rayos X.
   - Implementación inicial de un modelo de clasificación.

2. **Metas a Mediano Plazo:**
   - Evaluación y ajuste del modelo.
   - Validación cruzada y prueba en datos no vistos.

3. **Metas a Largo Plazo:**
   - Presentación y documentación del proyecto.
   - Publicación de resultados y recomendaciones.

Co-creación (lección 10) Desarrollo y Ajustes del Proyecto

Objetivos:
- Recolectar y preprocesar los datos necesarios para el proyecto.
- Implementar y evaluar el modelo de IA seleccionado.
- Ajustar y optimizar el modelo para mejorar su rendimiento.

Contenidos:

1. Recolección y Preprocesamiento de Datos

Objetivo: Recolectar y preparar los datos de imágenes de rayos X necesarios para el desarrollo del proyecto.

1. **Recolección de Datos:**
   - Obtener un conjunto de datos de imágenes de rayos X de fuentes públicas (por ejemplo, Kaggle).
   ```python
   import os
   import pandas as pd
   ```

```
data_dir = 'path_to_dataset'
images = [os.path.join(data_dir, img) for img in os.listdir(data_dir)]
labels = [1 if 'disease' in img else 0 for img in images]
df = pd.DataFrame({'image_path': images, 'label': labels})
print(df.head())
```

2. **Preprocesamiento de Imágenes:**
   - Redimensionamiento y normalización de imágenes.
   ```python
   from PIL import Image
   import numpy as np

   def preprocess_image(image_path):
       img = Image.open(image_path)
       img = img.resize((128, 128))
       img = np.array(img) / 255.0
       return img

   df['image'] = df['image_path'].apply(preprocess_image)
   print(df['image'].shape)
   ```

2. Implementación del Modelo de Clasificación

Objetivo: Desarrollar e implementar un modelo de clasificación de imágenes utilizando redes neuronales convolucionales (CNN).

1. **Selección del Modelo:**
   - Uso de una red neuronal convolucional (CNN) para la clasificación de imágenes.
   ```python
   from tensorflow.keras.models import Sequential
   from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

   model = Sequential([
       Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
       MaxPooling2D((2, 2)),
       Conv2D(64, (3, 3), activation='relu'),
       MaxPooling2D((2, 2)),
       Flatten(),
       Dense(128, activation='relu'),
       Dense(1, activation='sigmoid')
   ])

   model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
   print(model.summary())
   ```

2. **Entrenamiento del Modelo:**
   ```python
   from sklearn.model_selection import train_test_split

   X = np.stack(df['image'].values)
   y = df['label'].values
   X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

   model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
   ```

3. Evaluación y Ajuste del Modelo

Objetivo: Evaluar el rendimiento del modelo y realizar ajustes para mejorar su precisión.

1. **Evaluación del Modelo:**
   ```python
   loss, accuracy = model.evaluate(X_val, y_val)
   print(f'Validation Accuracy: {accuracy}')
   ```

2. **Ajuste de Hiperparámetros:**
   - Exploración de diferentes configuraciones de la red y parámetros de entrenamiento.
   ```python
   from tensorflow.keras.optimizers import Adam

   def build_model(learning_rate):
       model = Sequential([
           Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
           MaxPooling2D((2, 2)),
           Conv2D(64, (3, 3), activation='relu'),
           MaxPooling2D((2, 2)),
           Flatten(),
           Dense(128, activation='relu'),
           Dense(1, activation='sigmoid')
       ])
       model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
       return model

   model = build_model(learning_rate=0.001)
   model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
   ```

Co-creación y satélites (lección 11) Presentación, Retroalimentación y Networking del Proyecto

Objetivos:
- Preparar y presentar los resultados del proyecto.
- Discutir los hallazgos y recibir retroalimentación constructiva.

Contenidos:

1. Preparación de la Presentación

Objetivo: Preparar una presentación detallada para compartir los resultados del proyecto colaborativo.

1. **Estructura de la Presentación:**
   - Introducción al problema y objetivos del proyecto.
   - Descripción del conjunto de datos y el preprocesamiento realizado.
   - Implementación del modelo y resultados obtenidos.
   - Conclusiones y recomendaciones para futuros trabajos.

2. **Materiales de Apoyo:**
   - Diapositivas con gráficos y tablas.
   - Código fuente documentado y resultados visualizados.

2. Presentación del Proyecto

Objetivo: Presentar el proyecto final y los resultados obtenidos a compañeros y facilitadores.

1. **Exposición del Proyecto:**
   - Presentación del problema, metodología y resultados.
   - Discusión de los desafíos enfrentados y cómo se resolvieron.

2. **Sesión de Preguntas y Respuestas:**
   - Abrir espacio para preguntas y retroalimentación de los compañeros y facilitadores.

3. Discusión y Retroalimentación

Objetivo: Recibir retroalimentación constructiva y discutir posibles mejoras y futuros desarrollos del proyecto.

1. **Evaluación de Resultados:**
   - Análisis crítico de los resultados obtenidos y su significado.

2. **Retroalimentación:**
   - Recepción de sugerencias y comentarios para mejorar el proyecto.

3. **Planificación de Mejoras Futuras:**
   - Identificación de áreas de mejora y planificación de pasos futuros para el proyecto.

Talleres de Presentación de Proyectos y Networking

Objetivos:
- Presentar los proyectos finales de los participantes de manera profesional.
- Recibir retroalimentación constructiva de compañeros, facilitadores y profesionales invitados.
- Establecer conexiones con otros participantes y profesionales del sector a través de actividades de networking.

Contenidos:

1. Preparación para la Presentación

Objetivo: Asegurar que los participantes estén listos para presentar sus proyectos de manera clara y efectiva.

1. **Revisión de la Presentación:**
   - **Estructura de la Presentación:**
     - Introducción: Breve resumen del proyecto y los objetivos.
     - Metodología: Descripción del enfoque y las técnicas utilizadas.
     - Resultados: Presentación de los hallazgos y resultados del proyecto.
     - Conclusiones: Reflexiones finales y posibles mejoras futuras.
   - **Materiales de Apoyo:**
     - Diapositivas con gráficos, tablas y puntos clave.
     - Código fuente documentado y resultados visualizados.

2. **Práctica de la Presentación:**
   - Ensayo de la presentación para asegurar fluidez y claridad.
   - Preparación para responder posibles preguntas del público.
   - Asegurarse de que todo el equipo esté coordinado y listo para presentar.

2. Presentación del Proyecto Final

Objetivo: Permitir que cada equipo presente su proyecto, demostrando sus logros y el impacto de su trabajo.

1. **Configuración del Taller:**
   - **Moderador:** Designar a un moderador para introducir a cada equipo y gestionar el tiempo y las preguntas.
   - **Agenda:** Establecer un cronograma con tiempos asignados para cada presentación (por ejemplo, 10 minutos por equipo más 5 minutos para preguntas).

2. **Realización de Presentaciones:**

- **Introducción por el Moderador:** Breve introducción sobre el propósito del taller y la agenda del día.
- **Presentaciones de Equipos:** Cada equipo presenta su proyecto siguiendo la estructura preparada.
- **Sesión de Preguntas y Retroalimentación:** Después de cada presentación, abrir el espacio para preguntas del público y proporcionar retroalimentación constructiva.

3. **Evaluación y Retroalimentación:**
- **Criterios de Evaluación:** Evaluar cada presentación en términos de claridad, metodología, resultados y habilidades de presentación.
- **Comentarios Constructivos:** Proporcionar retroalimentación específica y constructiva para ayudar a los participantes a mejorar.

3. Actividades de Networking

Objetivo: Facilitar el networking entre los estudiantes, profesionales de la industria y potenciales empleadores, creando oportunidades para establecer conexiones valiosas y explorar futuras colaboraciones o oportunidades de empleo.

1. **Actividades de Networking:**
- **Speed Networking:** Organizar sesiones rápidas donde los estudiantes pueden interactuar con diferentes profesionales en intervalos cortos de tiempo.
- **Mesas Redondas Temáticas:** Crear mesas redondas enfocadas en temas específicos relacionados con la inteligencia artificial, donde los estudiantes pueden discutir y compartir ideas con expertos.
- **Intercambio de Contactos:** Proporcionar un espacio donde los estudiantes y los profesionales puedan intercambiar tarjetas de visita, perfiles de LinkedIn u otros medios de contacto.

2. **Consejos para el Networking:**
- **Presentación Personal:** Enseñar a los estudiantes cómo presentarse de manera efectiva en un entorno de networking.
- **Elevator Pitch:** Ayudar a los estudiantes a preparar un breve discurso que resuma quiénes son, qué hacen y qué buscan.
- **Seguimiento:** Aconsejar a los estudiantes sobre la importancia de hacer un seguimiento con las personas que conozcan durante el evento.

Proyecto (lección 12)

colocar una indicación referente a subir el proyecto final y una imagen

English Code (lección 13) Stylesheet language CSS

revisar documento drive estructura curso bootcmap, pero dejarlo para el final.

English Code (lección 14) Programming Language SQL and DataBase

revisar documento drive estructura curso bootcmap, pero dejarlo para el final.

Zona de recarga (Lección 15) Estrategias de Resolución de Problemas en Proyectos de IA

Objetivos:
- Desarrollar la capacidad de pensar críticamente y de manera analítica.
- Aplicar habilidades de pensamiento crítico para identificar y resolver problemas en proyectos de IA.
- Fomentar una actitud proactiva hacia la identificación y resolución de problemas.

Contenidos:

1. Introducción al Pensamiento Crítico y Resolución de Problemas

Objetivo: Comprender los fundamentos del pensamiento crítico y su importancia en el campo de la inteligencia artificial.

1. **Definición de Pensamiento Crítico:**
   - Qué es el pensamiento crítico.
   - Importancia del pensamiento crítico en la resolución de problemas complejos.

2. **Componentes del Pensamiento Crítico:**
   - Análisis.
   - Evaluación.
   - Inferencia.
   - Explicación.
   - Autorregulación.

3. **Características de un Pensador Crítico:**
   - Curiosidad.
   - Escepticismo saludable.
   - Humildad intelectual.
   - Perseverancia.

2. Aplicación del Pensamiento Crítico en Proyectos de IA

Objetivo: Aplicar habilidades de pensamiento crítico en la identificación y resolución de problemas específicos en proyectos de inteligencia artificial.

1. **Identificación de Problemas:**
   - Cómo identificar problemas en datos y modelos de IA.
   - Análisis de datos para detectar anomalías y patrones.
   ```python
   import pandas as pd
   ```

```
df = pd.read_csv('data.csv')
print(df.describe())
```

2. **Evaluación de Soluciones:**
   - Evaluar diferentes enfoques para resolver un problema.
   - Comparar ventajas y desventajas de cada enfoque.
   ```python
   from sklearn.model_selection import train_test_split
   from sklearn.linear_model import LogisticRegression
   from sklearn.tree import DecisionTreeClassifier

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
   log_reg = LogisticRegression().fit(X_train, y_train)
   tree_clf = DecisionTreeClassifier().fit(X_train, y_train)

   log_reg_accuracy = log_reg.score(X_test, y_test)
   tree_clf_accuracy = tree_clf.score(X_test, y_test)

   print(f'Logistic Regression Accuracy: {log_reg_accuracy}')
   print(f'Decision Tree Accuracy: {tree_clf_accuracy}')
   ```

3. **Solución de Problemas:**
   - Desarrollo de estrategias para resolver problemas identificados.
   - Implementación de soluciones efectivas.
   ```python
   # Solución de problema de datos faltantes
   df.fillna(df.mean(), inplace=True)
   ```

3. Estudios de Caso y Ejercicios Prácticos

Objetivo: Poner en práctica las habilidades de pensamiento crítico mediante estudios de caso y ejercicios prácticos.

1. **Estudio de Caso 1:**
   - Descripción de un problema de IA real (por ejemplo, detección de fraudes).
   - Análisis del problema y discusión de posibles soluciones.
   ```python
   # Análisis de transacciones bancarias para detección de fraudes
   df['fraud'] = (df['transaction_amount'] > 1000).astype(int)
   print(df.groupby('fraud').size())
   ```

2. **Estudio de Caso 2:**
   - Descripción de un problema de modelado predictivo (por ejemplo, predicción de ventas).
   - Evaluación de diferentes modelos y selección del mejor enfoque.

```python
# Predicción de ventas utilizando regresión lineal
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, y_train)
predictions = model.predict(X_test)
print(predictions)
```

3. **Ejercicios Prácticos:**
   - Resolución de problemas específicos utilizando pensamiento crítico.
   - Discusión en grupo de las soluciones propuestas.

Zona de recarga (Lección 16) Innovación y Creatividad en el Desarrollo de Aplicaciones de IA

1. Importancia de la Creatividad e Innovación en IA

Objetivo: Comprender la importancia de la creatividad y la innovación en el desarrollo de soluciones de IA.

1. **Beneficios de la Creatividad e Innovación:**
   - Diversidad de perspectivas y soluciones.
   - Mejora en la creatividad y la innovación.
   - Aumento de la eficiencia y la productividad.

2. **Desafíos de la Innovación:**
   - Coordinación de tareas y responsabilidades.
   - Gestión de conflictos y diferencias de opinión.
   - Comunicación efectiva en equipos multidisciplinarios.

2. Técnicas de Creatividad y Resolución de Problemas

Objetivo: Aprender y aplicar técnicas efectivas de creatividad y resolución de problemas en proyectos de IA.

1. **Metodologías de Creatividad:**
   - Brainstorming y lluvia de ideas.
   - Técnicas de pensamiento lateral.
   - Mapas mentales.
   ```python
   # Ejemplo de uso de diagramas de afinidad para agrupar ideas
   ideas = ["Mejora del modelo", "Nuevos datos", "Ajuste de hiperparámetros"]
   affinity_diagram = {"Model Improvement": ["Mejora del modelo", "Ajuste de hiperparámetros"], "New Data": ["Nuevos datos"]}
   print(affinity_diagram)
   ```

2. **Técnicas de Resolución de Problemas:**
   - Análisis de causa raíz (Diagrama de Ishikawa).
   - Técnicas de toma de decisiones (Matriz de Decisión).
   ```python
   # Ejemplo de análisis de causa raíz utilizando pandas
   df['root_cause'] = df['issue'].apply(lambda x: 'Data Issue' if 'data' in x else 'Model Issue')
   print(df.groupby('root_cause').size())
   ```

3. Ejercicios Prácticos y Dinámicas de Grupo

Objetivo: Poner en práctica las habilidades de creatividad y resolución de problemas mediante ejercicios prácticos y dinámicas de grupo.

1. **Ejercicio Práctico 1:**
   - Trabajar en equipo para resolver un problema de IA específico (por ejemplo, mejorar la precisión de un modelo).
   - Discusión de estrategias y soluciones propuestas.
   ```python
   # Mejora de la precisión del modelo mediante la optimización de hiperparámetros
   from sklearn.model_selection import GridSearchCV
   param_grid = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01]}
   grid = GridSearchCV(SVC(), param_grid, refit=True)
   grid.fit(X_train, y_train)
   print(grid.best_params_)
   ```

2. **Dinámica de Grupo:**
   - Simulación de una reunión de equipo para planificar y coordinar un proyecto de IA.
   - Resolución de conflictos y toma de decisiones colaborativa.
   ```python
   # Ejemplo de planificación de tareas utilizando un tablero Kanban
   kanban_board = {
       'To Do': ['Recolección de datos', 'Preprocesamiento'],
       'In Progress': ['Entrenamiento del modelo'],
       'Done': ['Exploración de datos']
   }
   print(kanban_board)
   ```

Misión 3: Construcción de Modelos de Machine Learning

Mapa de Ruta - Misión 3: Construcción de Modelos de Machine Learning

Mapa de Ruta

Objetivo General: Proporcionar a los participantes una comprensión profunda de los fundamentos del machine learning, así como las habilidades necesarias para desarrollar y evaluar modelos de machine learning. Los estudiantes aprenderán técnicas de preprocesamiento de datos, construcción de modelos, evaluación y optimización, culminando en la aplicación práctica a través de proyectos colaborativos.

Objetivos del Bootcamp

1. **Fundamentos de Machine Learning:**
   - Comprender los principios básicos del machine learning.
   - Conocer las diferentes técnicas y algoritmos utilizados en machine learning.

2. **Desarrollo de Modelos de Machine Learning:**
   - Aprender a construir y entrenar modelos de machine learning.
   - Aplicar técnicas de preprocesamiento de datos para preparar datos para el modelado.

3. **Evaluación y Optimización de Modelos:**
   - Evaluar el rendimiento de los modelos de machine learning.
   - Optimizar modelos para mejorar su precisión y eficiencia.

Habilidades Digitales a Desarrollar

1. **Construcción de Modelos de Machine Learning:**
   - Habilidad para construir y entrenar modelos utilizando diferentes algoritmos de machine learning.
   - Capacidad para aplicar técnicas de preprocesamiento y limpieza de datos.

2. **Evaluación y Mejora de Modelos:**
   - Competencia en la evaluación de modelos utilizando métricas de rendimiento.
   - Habilidad para optimizar modelos ajustando hiperparámetros y aplicando técnicas avanzadas.

3. **Desarrollo de Proyectos de Machine Learning:**
   - Experiencia en el desarrollo de proyectos prácticos de machine learning.
   - Capacidad para trabajar en equipo en proyectos colaborativos.

Contenidos a Abordar

1. **Fundamentos de Machine Learning:**
   - Principios básicos y tipos de machine learning (supervisado, no supervisado, semi-supervisado).
   - Algoritmos comunes de machine learning (regresión lineal, árboles de decisión, SVM, redes neuronales).

2. **Técnicas de Construcción de Modelos:**
   - Preprocesamiento y limpieza de datos.
   - Ingeniería de características.
   - Entrenamiento y ajuste de modelos.

3. **Evaluación y Optimización de Modelos:**
   - Métricas de evaluación (precisión, recall, F1-score, AUC-ROC).
   - Validación cruzada.
   - Optimización de hiperparámetros.

Propósitos Educativos del Bootcamp

- **Comprensión Profunda:** Proporcionar una comprensión sólida de los fundamentos y técnicas de machine learning.
- **Desarrollo de Habilidades:** Desarrollar habilidades prácticas en la construcción, evaluación y optimización de modelos de machine learning.
- **Aplicación de Conocimientos:** Aplicar los conocimientos adquiridos en proyectos prácticos que aborden problemas específicos en diversos sectores.

Alineación con el Mercado Laboral

El contenido del bootcamp está alineado con las necesidades del mercado laboral en el ámbito de la inteligencia artificial y el machine learning. Los estudiantes adquirirán competencias demandadas por la industria, incluyendo:

- Habilidades en la construcción y evaluación de modelos de machine learning.
- Experiencia práctica en el preprocesamiento de datos y la ingeniería de características.
- Capacidad para aplicar técnicas de machine learning en casos de uso reales.

Preparación para Desafíos Digitales

Los participantes estarán preparados para enfrentar desafíos digitales relacionados con el machine learning, incluyendo:

- Desarrollo de modelos de machine learning robustos y eficientes.
- Aplicación de técnicas avanzadas de preprocesamiento y optimización.
- Evaluación y mejora continua de modelos para asegurar su precisión y rendimiento.

Contenidos de la Misión 3

Entrenamiento

- **Mapa de Ruta:** Fundamentos de machine learning, desarrollo y evaluación de modelos.
  - **Objetivo:** Proporcionar una comprensión profunda de los fundamentos de machine learning y las técnicas de construcción y evaluación de modelos.
  - **Habilidades Digitales:** Construcción de modelos, preprocesamiento de datos, evaluación y optimización.
  - **Contenidos:**
    - Fundamentos y tipos de machine learning.
    - Técnicas de construcción y evaluación de modelos.
    - Preprocesamiento y limpieza de datos.

- **Preparación (15 horas):** Técnicas de construcción de modelos, preprocesamiento de datos y evaluación.
  - Introducción a los fundamentos de machine learning.
  - Preprocesamiento y limpieza de datos.
  - Construcción y entrenamiento de modelos.
  - Evaluación y optimización de modelos.

- **English Code (8 horas):** Terminología en inglés sobre machine learning y modelos de IA.
  - **Contenido:**
    - Vocabulario técnico en machine learning.
    - Lectura y comprensión de documentación técnica.
    - Redacción de comentarios y documentación en inglés.

Experiencia

- Construcción y evaluación de modelos de machine learning.
  - Configuración del entorno para el desarrollo de modelos.
  - Desarrollo y entrenamiento de un modelo de machine learning.
  - Evaluación y mejora del modelo.
  - Presentación y discusión de resultados.

- **Cápsulas:** Videos y tutoriales sobre machine learning y modelos de IA.
  - **Contenido:** Recursos multimedia que refuercen los conceptos aprendidos y muestren ejemplos prácticos.

- **Zona de Recarga (5 horas):** Mejores prácticas en la construcción de modelos y gestión de proyectos.
  - Estrategias de construcción de modelos eficientes.
  - Gestión de proyectos de machine learning.

Conexión

- **Prosumidores (3 horas):** Desarrollo de un pequeño proyecto de machine learning.
  - **Contenido:** Desarrollo de un proyecto práctico que aplique técnicas de machine learning en un contexto real.

- **Co-creación (10 horas):** Proyecto colaborativo enfocado en la construcción de modelos de IA para un sector específico.
  - Definición del proyecto colaborativo.
  - Desarrollo y ajustes del proyecto.
  - Presentación del proyecto y retroalimentación.

- **Satélites (2 horas):** Talleres de presentación de proyectos y networking.
  - Taller de presentación de proyectos y actividades de networking.

Preparación (lección 1) Introducción a los Fundamentos de Machine Learning

Objetivos:
- Comprender los principios básicos del machine learning.
- Familiarizarse con los tipos de machine learning y sus aplicaciones.
- Conocer los algoritmos más comunes utilizados en machine learning.

Contenidos:

1. Fundamentos de Machine Learning

Objetivo: Proporcionar una comprensión básica de qué es el machine learning y cómo se aplica en diversos contextos.

1. **Definición de Machine Learning:**
   - Qué es el machine learning.
   - Diferencia entre machine learning y programación tradicional.
   - Aplicaciones del machine learning en la vida cotidiana.

2. **Tipos de Machine Learning:**
   - Aprendizaje Supervisado.
   - Aprendizaje No Supervisado.
   - Aprendizaje Semi-supervisado.
   - Aprendizaje por Refuerzo.

3. **Ciclo de Vida de un Proyecto de Machine Learning:**
   - Definición del problema.
   - Recolección y preparación de datos.
   - Selección de un modelo.
   - Entrenamiento del modelo.
   - Evaluación y validación del modelo.
   - Despliegue y monitoreo del modelo.

2. Algoritmos Comunes de Machine Learning

Objetivo: Conocer los algoritmos más utilizados en machine learning y sus aplicaciones.

1. **Regresión Lineal:**
   - Conceptos básicos.
   - Ejemplo de aplicación.
   ```python
   from sklearn.linear_model import LinearRegression
   model = LinearRegression()
   model.fit(X_train, y_train)
   predictions = model.predict(X_test)
   ```

2. **Árboles de Decisión:**
   - Conceptos básicos.
   - Ejemplo de aplicación.
   ```python
   from sklearn.tree import DecisionTreeClassifier
   model = DecisionTreeClassifier()
   model.fit(X_train, y_train)
   predictions = model.predict(X_test)
   ```

3. **Máquinas de Soporte Vectorial (SVM):**
   - Conceptos básicos.
   - Ejemplo de aplicación.
   ```python
   from sklearn.svm import SVC
   model = SVC()
   model.fit(X_train, y_train)
   predictions = model.predict(X_test)
   ```

4. **Redes Neuronales:**
   - Conceptos básicos.
   - Ejemplo de aplicación.
   ```python
   from tensorflow.keras.models import Sequential
   from tensorflow.keras.layers import Dense

   model = Sequential()
   model.add(Dense(12, input_dim=8, activation='relu'))
   model.add(Dense(8, activation='relu'))
   model.add(Dense(1, activation='sigmoid'))

   model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
   model.fit(X_train, y_train, epochs=150, batch_size=10)
   predictions = model.predict(X_test)
   ```

3. Aplicaciones Prácticas de Machine Learning

Objetivo: Ver cómo se utilizan los algoritmos de machine learning en la práctica a través de estudios de caso.

1. **Estudio de Caso 1: Predicción de Precios de Viviendas:**
   - Descripción del problema.
   - Datos utilizados.
   - Algoritmo aplicado y resultados obtenidos.

2. **Estudio de Caso 2: Detección de Fraudes en Transacciones Bancarias:**
   - Descripción del problema.
   - Datos utilizados.
   - Algoritmo aplicado y resultados obtenidos.

3. **Estudio de Caso 3: Clasificación de Imágenes:**
   - Descripción del problema.
   - Datos utilizados.
   - Algoritmo aplicado y resultados obtenidos.

Preparación (lección 2) Preprocesamiento y Limpieza de Datos

Objetivos:
- Comprender la importancia del preprocesamiento de datos en machine learning.
- Aprender técnicas de limpieza y transformación de datos.
- Aplicar técnicas de ingeniería de características.

Contenidos:

1. Importancia del Preprocesamiento de Datos

Objetivo: Entender por qué el preprocesamiento de datos es crucial para el éxito de los modelos de machine learning.

1. **Calidad de los Datos:**
   - Importancia de la calidad de los datos.
   - Impacto de los datos sucios en el rendimiento del modelo.

2. **Pasos en el Preprocesamiento de Datos:**
   - Recolección de datos.
   - Limpieza de datos.
   - Transformación de datos.
   - Ingeniería de características.

2. Limpieza de Datos

Objetivo: Aprender y aplicar técnicas para limpiar los datos.

1. **Manejo de Datos Faltantes:**
   - Identificación de datos faltantes.
   - Métodos para manejar datos faltantes (eliminación, imputación).
   ```python
   df.dropna(inplace=True)
   df.fillna(df.mean(), inplace=True)
   ```

2. **Eliminación de Duplicados:**
   - Identificación y eliminación de registros duplicados.
   ```python
   df.drop_duplicates(inplace=True)
   ```

3. **Corrección de Valores Erróneos:**
   - Detección y corrección de valores atípicos y errores en los datos.
   ```python
   df.loc[df['age'] > 100, 'age'] = df['age'].median()
   ```

3. Transformación de Datos

Objetivo: Aprender y aplicar técnicas de transformación de datos para preparar los datos para el modelado.

1. **Normalización y Estandarización:**
   - Conceptos de normalización y estandarización.
   - Aplicación en Python.
   ```python
   from sklearn.preprocessing import StandardScaler, MinMaxScaler

   scaler = StandardScaler()
   df_scaled = scaler.fit_transform(df)

   minmax_scaler = MinMaxScaler()
   df_normalized = minmax_scaler.fit_transform(df)
   ```

2. **Codificación de Variables Categóricas:**
   - Métodos de codificación (one-hot encoding, label encoding).
   - Aplicación en Python.
   ```python
   from sklearn.preprocessing import OneHotEncoder, LabelEncoder

   label_encoder = LabelEncoder()
   df['category_encoded'] = label_encoder.fit_transform(df['category'])

   onehot_encoder = OneHotEncoder()
   df_onehot = onehot_encoder.fit_transform(df[['category']])
   ```

3. **Transformaciones Logarítmicas y de Potencia:**
   - Cuándo y cómo aplicar transformaciones logarítmicas y de potencia.
   ```python
   import numpy as np
   ```

```
df['log_transformed'] = np.log(df['variable'])
df['sqrt_transformed'] = np.sqrt(df['variable'])
```

4. Ingeniería de Características

Objetivo: Aprender técnicas de ingeniería de características para mejorar el rendimiento del modelo.

1. **Selección de Características:**
   - Métodos de selección de características (selección univariante, selección basada en modelo).
   ```python
   from sklearn.feature_selection import SelectKBest, f_classif

   selector = SelectKBest(score_func=f_classif, k=5)
   X_new = selector.fit_transform(X, y)
   ```

2. **Creación de Nuevas Características:**
   - Técnicas para crear nuevas características a partir de las existentes.
   ```python
   df['interaction'] = df['feature1'] * df['feature2']
   ```

3. **Transformaciones de Características:**
   - Métodos para transformar características para mejorar el rendimiento del modelo.
   ```python
   df['binned'] = pd.cut(df['variable'], bins=5, labels=False)
   ```

Preparación (lección 3) Construcción y Entrenamiento de Modelos

1. Selección de Algoritmos de Machine Learning

Objetivo: Entender cómo seleccionar el algoritmo de machine learning adecuado para un problema específico.

1. **Criterios de Selección:**
   - Tipo de problema (regresión, clasificación).
   - Tamaño y calidad del conjunto de datos.
   - Interpretabilidad del modelo.

2. **Comparación de Algoritmos:**
   - Ventajas y desventajas de diferentes algoritmos.
   - Casos de uso comunes para cada algoritmo.

2. Construcción y Entrenamiento de Modelos

Objetivo: Aprender a construir y entrenar modelos de machine learning utilizando diferentes algoritmos.

1. **Regresión Lineal:**
   - Construcción y entrenamiento del modelo.
   ```python
   from sklearn.linear_model import LinearRegression
   model = LinearRegression()
   model.fit(X_train, y_train)
   predictions = model.predict(X_test)
   ```

2. **Árboles de Decisión:**
   - Construcción y entrenamiento del modelo.
   ```python
   from sklearn.tree import DecisionTreeClassifier
   model = DecisionTreeClassifier()
   model.fit(X_train, y_train)
   predictions =

  model.predict(X_test)
   ```

3. **Máquinas de Soporte Vectorial (SVM):**
   - Construcción y entrenamiento del modelo.
   ```python
   from sklearn.svm import SVC
   model = SVC()
   model.fit(X_train, y_train)
   predictions = model.predict(X_test)
   ```

4. **Redes Neuronales:**
   - Construcción y entrenamiento del modelo.
   ```python
   from tensorflow.keras.models import Sequential
   from tensorflow.keras.layers import Dense

   model = Sequential()
   model.add(Dense(12, input_dim=8, activation='relu'))
   model.add(Dense(8, activation='relu'))
   model.add(Dense(1, activation='sigmoid'))

   model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
   model.fit(X_train, y_train, epochs=150, batch_size=10)
   predictions = model.predict(X_test)
   ```

```

#### 3. Evaluación del Rendimiento del Modelo

**Objetivo:** Aprender a evaluar el rendimiento de los modelos utilizando métricas apropiadas y técnicas de validación.

1. **Métricas de Evaluación:**
   - Precisión, recall, F1-score, AUC-ROC.
   ```python
   from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

   accuracy = accuracy_score(y_test, predictions)
   precision = precision_score(y_test, predictions)
   recall = recall_score(y_test, predictions)
   f1 = f1_score(y_test, predictions)
   auc = roc_auc_score(y_test, predictions)

   print(f'Accuracy: {accuracy}')
   print(f'Precision: {precision}')
   print(f'Recall: {recall}')
   print(f'F1 Score: {f1}')
   print(f'AUC-ROC: {auc}')
   ```

2. **Validación Cruzada:**
   - Aplicación de técnicas de validación cruzada para evaluar el rendimiento del modelo.
   ```python
   from sklearn.model_selection import cross_val_score

   scores = cross_val_score(model, X, y, cv=5)
   print(f'Cross-Validation Scores: {scores}')
   print(f'Mean Score: {scores.mean()}')
   ```

3. **Ajuste de Hiperparámetros:**
   - Técnicas para ajustar los hiperparámetros del modelo para mejorar su rendimiento.
   ```python
   from sklearn.model_selection import GridSearchCV

   param_grid = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01]}
   grid_search = GridSearchCV(SVC(), param_grid, cv=5)
   grid_search.fit(X_train, y_train)

   print(f'Best Parameters: {grid_search.best_params_}')
   best_model = grid_search.best_estimator_
   best_predictions = best_model.predict(X_test)
```
```

```

## Preparación y simulación (lección 4) Evaluación, Optimización y Configuración del Entorno para el Desarrollo de Modelos

#### 1. Evaluación Avanzada del Rendimiento del Modelo

**Objetivo:** Aplicar técnicas avanzadas para evaluar el rendimiento del modelo y entender sus limitaciones.

1. **Análisis de Error:**
   - Identificación y análisis de errores en las predicciones del modelo.
   ```python
   from sklearn.metrics import confusion_matrix

   cm = confusion_matrix(y_test, predictions)
   print(cm)
   ```

2. **Curvas de Aprendizaje:**
   - Uso de curvas de aprendizaje para evaluar el rendimiento del modelo a lo largo del tiempo.
   ```python
   from sklearn.model_selection import learning_curve
   import matplotlib.pyplot as plt

   train_sizes, train_scores, test_scores = learning_curve(model, X, y, cv=5)
   plt.plot(train_sizes, train_scores.mean(axis=1), label='Train score')
   plt.plot(train_sizes, test_scores.mean(axis=1), label='Test score')
   plt.xlabel('Training examples')
   plt.ylabel('Score')
   plt.legend()
   plt.show()
   ```

3. **Evaluación de Sensibilidad:**
   - Análisis de la sensibilidad del modelo a diferentes características y parámetros.
   ```python
   import numpy as np

   def evaluate_sensitivity(model, X, y):
       sensitivities = []
       for feature in range(X.shape[1]):
           X_temp = np.copy(X)
           X_temp[:, feature] = np.random.permutation(X_temp[:, feature])
           score = cross_val_score(model, X_temp, y, cv=5).mean()
           sensitivities.append(score)
```
```

```
    return sensitivities

sensitivities = evaluate_sensitivity(model, X, y)
print(sensitivities)
```

2. Optimización del Modelo

Objetivo: Implementar técnicas avanzadas para optimizar el rendimiento del modelo de machine learning.

1. **Ajuste Fino del Modelo:**
   - Técnicas de ajuste fino para mejorar la precisión y eficiencia del modelo.
   ```python
   from sklearn.model_selection import RandomizedSearchCV

   param_dist = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01]}
   random_search = RandomizedSearchCV(SVC(), param_dist, cv=5, n_iter=10)
   random_search.fit(X_train, y_train)

   print(f'Best Parameters: {random_search.best_params_}')
   best_model = random_search.best_estimator_
   best_predictions = best_model.predict(X_test)
   ```

2. **Ensemble Learning:**
   - Aplicación de técnicas de ensemble learning para mejorar el rendimiento del modelo.
   ```python
   from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

   rf_model = RandomForestClassifier()
   gb_model = GradientBoostingClassifier()

   rf_model.fit(X_train, y_train)
   gb_model.fit(X_train, y_train)

   rf_predictions = rf_model.predict(X_test)
   gb_predictions = gb_model.predict(X_test)

   print(f'Random Forest Accuracy: {accuracy_score(y_test, rf_predictions)}')
   print(f'Gradient Boosting Accuracy: {accuracy_score(y_test, gb_predictions)}')
   ```

3. **Regularización:**
   - Uso de técnicas de regularización para prevenir el sobreajuste y mejorar la generalización del modelo.
   ```python
   from sklearn.linear_model import Ridge, Lasso
```

```
ridge_model = Ridge(alpha=1.0)
lasso_model = Lasso(alpha=0.1)

ridge_model.fit(X_train, y_train)
lasso_model.fit(X_train, y_train)

ridge_predictions = ridge_model.predict(X_test)
lasso_predictions = lasso_model.predict(X_test)

print(f'Ridge Regression Accuracy: {ridge_model.score(X_test, y_test)}')
print(f'Lasso Regression Accuracy: {lasso_model.score(X_test, y_test)}')
```

Configuración del Entorno para el Desarrollo de Modelos

Objetivos:
- Configurar el entorno de desarrollo necesario para la construcción y evaluación de modelos de machine learning.
- Instalar y verificar las bibliotecas y herramientas requeridas.

Contenidos:

1. Instalación de Herramientas y Bibliotecas

Objetivo: Asegurar que todos los participantes tengan el entorno de desarrollo correctamente configurado.

1. **Instalación de Python y Bibliotecas Esenciales:**
   - Instalación de Python (si no está ya instalado).
   - Instalación de bibliotecas necesarias: scikit-learn, pandas, numpy, matplotlib, seaborn.
   ```bash
   pip install scikit-learn pandas numpy matplotlib seaborn
   ```

2. **Configuración del Entorno de Desarrollo:**
   - Uso de Jupyter Notebook o Visual Studio Code para el desarrollo.
   - Instalación de extensiones relevantes para la codificación en Python.

2. Verificación de la Configuración

Objetivo: Verificar que todas las instalaciones y configuraciones estén funcionando correctamente.

1. **Prueba de Instalación de scikit-learn:**
   - Importar y utilizar scikit-learn en un pequeño script de prueba.
   ```python
   import sklearn
   ```

```
print(sklearn.__version__)
```

2. **Configuración de un Notebook de Jupyter:**
   - Crear y ejecutar un notebook de Jupyter para probar la configuración.
   ```python
   import pandas as pd
   import numpy as np

   # Crear un pequeño DataFrame de prueba
   df = pd.DataFrame({
       'feature1': np.random.rand(10),
       'feature2': np.random.rand(10),
       'target': np.random.randint(0, 2, size=10)
   })
   print(df.head())
   ```

Simulación (lección 5) Desarrollo y Entrenamiento de un Modelo de Machine Learning

Objetivos:
- Desarrollar y entrenar un modelo de machine learning utilizando un conjunto de datos real.
- Aplicar técnicas de preprocesamiento de datos para preparar los datos para el modelado.
- Evaluar el rendimiento inicial del modelo utilizando métricas apropiadas.

Contenidos:

1. Recolección y Preprocesamiento de Datos

Objetivo: Recolectar y preparar los datos necesarios para el desarrollo del modelo.

1. **Recolección de Datos:**
   - Uso de un conjunto de datos público (por ejemplo, el conjunto de datos de Titanic de Kaggle).
   ```python
   import pandas as pd

   # Cargar el conjunto de datos de Titanic
   url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
   df = pd.read_csv(url)
   print(df.head())
   ```

2. **Preprocesamiento de Datos:**
   - Limpieza y transformación de datos para el modelado.
   ```python
   # Eliminar columnas innecesarias
   df.drop(columns=['Name', 'Ticket', 'Cabin'], inplace=True)

   # Manejo de datos faltantes
   df['Age'].fillna(df['Age'].mean(), inplace=True)
   df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

   # Codificación de variables categóricas
   df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)
   print(df.head())
   ```

3. **División de Datos:**
   - Dividir el conjunto de datos en conjuntos de entrenamiento y prueba.
   ```python
   from sklearn.model_selection import train_test_split

   X = df.drop(columns=['Survived'])
   y = df['Survived']

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
   print(X_train.shape, X_test.shape)
   ```

2. Construcción y Entrenamiento del Modelo

Objetivo: Construir y entrenar un modelo de machine learning utilizando el conjunto de datos preparado.

1. **Selección del Algoritmo:**
   - Seleccionar un algoritmo de clasificación adecuado (por ejemplo, Random Forest).
   ```python
   from sklearn.ensemble import RandomForestClassifier

   model = RandomForestClassifier(n_estimators=100, random_state=42)
   model.fit(X_train, y_train)
   ```

2. **Entrenamiento del Modelo:**
   - Entrenar el modelo utilizando el conjunto de entrenamiento.
   ```python
   model.fit(X_train, y_train)
   ```

3. **Evaluación Inicial del Modelo:**

- Evaluar el rendimiento del modelo utilizando el conjunto de prueba.
```python
from sklearn.metrics import accuracy_score, classification_report

y_pred = model.predict(X_test)
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(classification_report(y_test, y_pred))
```

3. Visualización de Resultados

Objetivo: Visualizar los resultados del modelo y analizar su rendimiento.

1. **Matriz de Confusión:**
   - Visualizar la matriz de confusión para entender mejor los resultados del modelo.
   ```python
   from sklearn.metrics import confusion_matrix
   import seaborn as sns
   import matplotlib.pyplot as plt

   cm = confusion_matrix(y_test, y_pred)
   sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
   plt.xlabel('Predicted')
   plt.ylabel('Actual')
   plt.show()
   ```

2. **Curvas ROC y AUC:**
   - Visualizar la curva ROC y calcular el AUC para evaluar el rendimiento del modelo.
   ```python
   from sklearn.metrics import roc_curve, roc_auc_score

   y_proba = model.predict_proba(X_test)[:, 1]
   fpr, tpr, _ = roc_curve(y_test, y_proba)
   auc = roc_auc_score(y_test, y_proba)

   plt.plot(fpr, tpr, label=f'AUC = {auc:.2f}')
   plt.xlabel('False Positive Rate')
   plt.ylabel('True Positive Rate')
   plt.title('ROC Curve')
   plt.legend(loc='best')
   plt.show()
   ```

Simulación (lección 6) Evaluación y Mejora del Modelo

Objetivos:

- Evaluar el rendimiento del modelo utilizando técnicas avanzadas de validación.
- Aplicar técnicas de optimización para mejorar el rendimiento del modelo.
- Documentar y presentar los resultados obtenidos.

Contenidos:

1. Validación del Modelo

Objetivo: Evaluar el rendimiento del modelo utilizando técnicas de validación cruzada.

1. **Validación Cruzada:**
   - Aplicar validación cruzada para evaluar el rendimiento del modelo de manera más robusta.
   ```python
   from sklearn.model_selection import cross_val_score

   scores = cross_val_score(model, X, y, cv=5)
   print(f'Cross-Validation Scores: {scores}')
   print(f'Mean Score: {scores.mean()}')
   ```

2. **Análisis de Resultados:**
   - Analizar los resultados de la validación cruzada y comparar con la evaluación inicial.

2. Optimización del Modelo

Objetivo: Aplicar técnicas de optimización para mejorar el rendimiento del modelo.

1. **Optimización de Hiperparámetros:**
   - Utilizar GridSearchCV para encontrar los mejores hiperparámetros para el modelo.
   ```python
   from sklearn.model_selection import GridSearchCV

   param_grid = {
       'n_estimators': [50, 100, 200],
       'max_depth': [None, 10, 20, 30],
       'min_samples_split': [2, 5, 10],
       'min_samples_leaf': [1, 2, 4]
   }
   grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
   grid_search.fit(X_train, y_train)
   print(f'Best Parameters: {grid_search.best_params_}')
   best_model = grid_search.best_estimator_
   ```

2. **Reentrenamiento del Modelo:**
   - Entrenar el modelo optimizado utilizando los mejores hiperparámetros.

```python
best_model.fit(X_train, y_train)
```

3. **Evaluación del Modelo Optimizado:**
   - Evaluar el rendimiento del modelo optimizado utilizando el conjunto de prueba.
   ```python
   y_pred_optimized = best_model.predict(X_test)
   print(f'Optimized Accuracy: {accuracy_score(y_test, y_pred_optimized)}')
   print(classification_report(y_test, y_pred_optimized))
   ```

3. Documentación y Presentación de Resultados

Objetivo: Documentar el proceso de desarrollo y presentar los resultados obtenidos.

1. **Documentación del Proceso:**
   - Redacción de un informe detallado sobre el desarrollo, entrenamiento y optimización del modelo.
   ```markdown
   # Desarrollo de un Modelo de Machine Learning

   ## Introducción
   Descripción del problema y objetivos del proyecto.

   ## Recolección y Preprocesamiento de Datos


   Detalles sobre la recolección y preparación de los datos.

   ## Construcción y Entrenamiento del Modelo
   Descripción de los algoritmos utilizados y resultados iniciales.

   ## Optimización del Modelo
   Técnicas de optimización aplicadas y resultados obtenidos.

   ## Conclusiones
   Reflexiones finales y posibles mejoras futuras.
   ```

2. **Preparación de la Presentación:**
   - Creación de diapositivas para presentar el desarrollo y los resultados del modelo.
   - Incluir gráficos y tablas para ilustrar los resultados obtenidos.

Simulación y prosumidor (lección 7) Presentación y Discusión de Resultados y Desarrollo de un Pequeño Proyecto de Machine Learning

Objetivos:
- Presentar los resultados del proyecto de construcción y evaluación de modelos de machine learning.
- Recibir retroalimentación y discutir posibles mejoras.

Contenidos:

1. Presentación del Proyecto

Objetivo: Compartir los hallazgos y resultados obtenidos durante el desarrollo del modelo.

1. **Estructura de la Presentación:**
   - Introducción al problema y objetivos del proyecto.
   - Descripción del conjunto de datos y el preprocesamiento realizado.
   - Implementación del modelo y resultados obtenidos.
   - Optimización del modelo y resultados finales.
   - Conclusiones y recomendaciones futuras.

2. **Materiales de Apoyo:**
   - Diapositivas con gráficos y tablas.
   - Código fuente documentado y resultados visualizados.

2. Discusión y Retroalimentación

Objetivo: Recibir retroalimentación constructiva y discutir posibles mejoras y futuros desarrollos del proyecto.

1. **Evaluación de Resultados:**
   - Análisis crítico de los resultados obtenidos y su significado.

2. **Retroalimentación:**
   - Recepción de sugerencias y comentarios para mejorar el proyecto.

3. **Planificación de Mejoras Futuras:**
   - Identificación de áreas de mejora y planificación de pasos futuros para el proyecto.

Desarrollo de un Pequeño Proyecto de Machine Learning

Objetivos:
- Identificar un problema específico que pueda ser resuelto utilizando técnicas de machine learning.

- Desarrollar y entrenar un modelo de machine learning para abordar el problema seleccionado.
- Evaluar y presentar los resultados del proyecto.

Contenidos:

1. Selección y Definición del Proyecto

Objetivo: Identificar un problema específico y definir el alcance del proyecto de machine learning.

1. **Selección del Problema:**
   - Realizar un brainstorming para identificar posibles problemas que se puedan resolver con machine learning.
   - Ejemplos de problemas simples:
     - Predicción de precios de viviendas.
     - Clasificación de correos electrónicos como spam o no spam.
     - Predicción de enfermedades a partir de datos médicos.

2. **Definición del Proyecto:**
   - **Título del Proyecto:** Clasificación de correos electrónicos como spam o no spam.
   - **Descripción del Problema:** Desarrollar un modelo de machine learning que clasifique correos electrónicos en dos categorías: spam y no spam.
   - **Objetivos del Proyecto:**
     - Recolectar y preparar un conjunto de datos de correos electrónicos.
     - Implementar un algoritmo de clasificación (por ejemplo, Naive Bayes).
     - Evaluar el rendimiento del modelo.

2. Recolección y Preprocesamiento de Datos

Objetivo: Recolectar y preparar los datos necesarios para el proyecto de machine learning.

1. **Recolección de Datos:**
   - Utilizar un conjunto de datos público de correos electrónicos (por ejemplo, el dataset de spam de Enron).
   ```python
   import pandas as pd

   # Cargar el conjunto de datos de correos electrónicos
   url = 'https://raw.githubusercontent.com/selva86/datasets/master/SMSSpamCollection'
   df = pd.read_csv(url, sep='\t', header=None, names=['label', 'message'])
   print(df.head())
   ```

2. **Preprocesamiento de Datos:**
   - Limpieza y transformación de los datos para el modelado.
   ```python
   ```

```
import re
from nltk.corpus import stopwords

def preprocess_text(text):
    text = re.sub(r'\W', ' ', text)
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    return text

df['message'] = df['message'].apply(preprocess_text)
stop_words = set(stopwords.words('english'))
df['message'] = df['message'].apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))
print(df.head())
```

3. **División de Datos:**
   - Dividir el conjunto de datos en conjuntos de entrenamiento y prueba.
   ```python
   from sklearn.model_selection import train_test_split

   X = df['message']
   y = df['label'].apply(lambda x: 1 if x == 'spam' else 0)

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
   print(X_train.shape, X_test.shape)
   ```

3. Construcción y Entrenamiento del Modelo

Objetivo: Construir y entrenar un modelo de machine learning utilizando el conjunto de datos preparado.

1. **Vectorización del Texto:**
   - Convertir el texto en vectores numéricos utilizando TF-IDF.
   ```python
   from sklearn.feature_extraction.text import TfidfVectorizer

   vectorizer = TfidfVectorizer(max_features=5000)
   X_train_tfidf = vectorizer.fit_transform(X_train)
   X_test_tfidf = vectorizer.transform(X_test)
   ```

2. **Entrenamiento del Modelo:**
   - Entrenar un modelo de Naive Bayes utilizando los datos vectorizados.
   ```python
   from sklearn.naive_bayes import MultinomialNB
   ```

```
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
```

3. **Evaluación Inicial del Modelo:**
   - Evaluar el rendimiento del modelo utilizando el conjunto de prueba.
   ```python
   from sklearn.metrics import accuracy_score, classification_report

   y_pred = model.predict(X_test_tfidf)
   print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
   print(classification_report(y_test, y_pred))
   ```

4. Optimización y Evaluación del Modelo

Objetivo: Optimizar el modelo y evaluar su rendimiento.

1. **Ajuste de Hiperparámetros:**
   - Explorar diferentes valores de hiperparámetros para mejorar el rendimiento del modelo.
   ```python
   from sklearn.model_selection import GridSearchCV

   param_grid = {'alpha': [0.1, 0.5, 1.0]}
   grid_search = GridSearchCV(MultinomialNB(), param_grid, cv=5)
   grid_search.fit(X_train_tfidf, y_train)

   print(f'Best Parameters: {grid_search.best_params_}')
   best_model = grid_search.best_estimator_
   ```

2. **Reentrenamiento y Evaluación del Modelo Optimizado:**
   - Entrenar el modelo optimizado y evaluar su rendimiento.
   ```python
   best_model.fit(X_train_tfidf, y_train)
   y_pred_optimized = best_model.predict(X_test_tfidf)
   print(f'Optimized Accuracy: {accuracy_score(y_test, y_pred_optimized)}')
   print(classification_report(y_test, y_pred_optimized))
   ```

5. Documentación y Presentación de Resultados

Objetivo: Documentar el proceso de desarrollo y presentar los resultados obtenidos.

1. **Documentación del Proceso:**
   - Redacción de un informe detallado sobre el desarrollo, entrenamiento y optimización del modelo.
   ```markdown
   ```

```
# Desarrollo de un Modelo de Clasificación de Correos Electrónicos como Spam

## Introducción
Descripción del problema y objetivos del proyecto.

## Recolección y Preprocesamiento de Datos
Detalles sobre la recolección y preparación de los datos.

## Construcción y Entrenamiento del Modelo
Descripción de los algoritmos utilizados y resultados iniciales.

## Optimización del Modelo
Técnicas de optimización aplicadas y resultados obtenidos.

## Conclusiones
Reflexiones finales y posibles mejoras futuras.
```

2. **Preparación de la Presentación:**
   - Creación de diapositivas para presentar el desarrollo y los resultados del modelo.
   - Incluir gráficos y tablas para ilustrar los resultados obtenidos.

3. **Presentación del Proyecto:**
   - Exposición del proyecto, metodología y resultados obtenidos.
   - Recepción de preguntas y retroalimentación de los compañeros y facilitadores.

Cápsulas (lección 8)

Videos y tutoriales sobre machine learning y modelos de IA

Links de recursos según competencia especificas curados (yudi)

Co-creación (lección 9) Definición del Proyecto Colaborativo

Objetivos:
- Seleccionar un sector específico y definir el problema a resolver.
- Planificar el proyecto colaborativo, asignar roles y responsabilidades.
- Establecer metas y cronograma para el desarrollo del proyecto.

Contenidos:

1. Selección del Sector Específico y Problema a Resolver

Objetivo: Identificar un sector específico y definir un problema práctico que pueda ser resuelto utilizando técnicas de IA.

1. **Brainstorming de Sectores y Problemas:**
   - Educación: Predicción de rendimiento estudiantil.
   - Salud: Diagnóstico de enfermedades a partir de imágenes médicas.
   - Finanzas: Detección de fraudes en transacciones bancarias.
   - Marketing: Análisis de sentimientos en redes sociales.
   - Agricultura: Predicción de rendimientos de cultivos.

2. **Definición del Problema:**
   - **Sector Seleccionado:** Salud.
   - **Problema a Resolver:** Diagnóstico de enfermedades pulmonares a partir de imágenes de rayos X.
   - **Objetivo del Proyecto:** Desarrollar un modelo de IA que clasifique imágenes de rayos X en categorías de enfermedades pulmonares.

2. Planificación del Proyecto

Objetivo: Planificar el proyecto colaborativo y asignar roles y responsabilidades a los miembros del equipo.

1. **Definición del Alcance:**
   - Especificar qué se incluirá y qué no se incluirá en el proyecto.
   - Establecer criterios de éxito para el proyecto.

2. **Asignación de Roles y Responsabilidades:**
   - Project Manager: Coordinación y supervisión general.
   - Data Scientists: Recolección y preprocesamiento de datos.
   - Machine Learning Engineers: Desarrollo e implementación del modelo.
   - Data Analysts: Evaluación y análisis de resultados.

3. **Desarrollo de un Cronograma:**
   - Dividir el proyecto en fases y establecer plazos para cada fase.
   - Crear un calendario de reuniones y revisiones de progreso.

3. Establecimiento de Metas y Resultados Esperados

Objetivo: Definir metas a corto, mediano y largo plazo para el proyecto, y establecer resultados esperados.

1. **Metas a Corto Plazo:**
   - Recolección y preprocesamiento de datos de imágenes de rayos X.
   - Implementación inicial de un modelo de clasificación.

2. **Metas a Mediano Plazo:**
   - Evaluación y ajuste del modelo.
   - Validación cruzada y prueba en datos no vistos.

3. **Metas a Largo Plazo:**

- Presentación y documentación del proyecto.
- Publicación de resultados y recomendaciones.

Co-creación (lección 10) Desarrollo y Ajustes del Proyecto

1. Recolección y Preprocesamiento de Datos

Objetivo: Recolectar y preparar los datos de imágenes de rayos X necesarios para el desarrollo del proyecto.

1. **Recolección de Datos:**
   - Obtener un conjunto de datos de imágenes de rayos X de fuentes públicas (por ejemplo, Kaggle).
   ```python
   import os
   import pandas as pd

   data_dir = 'path_to_dataset'
   images = [os.path.join(data_dir, img) for img in os.listdir(data_dir)]
   labels = [1 if 'disease' in img else 0 for img in images]
   df = pd.DataFrame({'image_path': images, 'label': labels})
   print(df.head())
   ```

2. **Preprocesamiento de Imágenes:**
   - Redimensionamiento y normalización de imágenes.
   ```python
   from PIL import Image
   import numpy as np

   def preprocess_image(image_path):
       img = Image.open(image_path)
       img = img.resize((128, 128))
       img = np.array(img) / 255.0
       return img

   df['image'] = df['image_path'].apply(preprocess_image)
   print(df['image'].shape)
   ```

2. Implementación del Modelo de Clasificación

Objetivo: Desarrollar e implementar un modelo de clasificación de imágenes utilizando redes neuronales convolucionales (CNN).

1. **Selección del Modelo:**
   - Uso de una red neuronal convolucional (CNN) para la clasificación de imágenes.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print(model.summary())
```

2. **Entrenamiento del Modelo:**
```python
from sklearn.model_selection import train_test_split

X = np.stack(df['image'].values)
y = df['label'].values
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
```

3. Evaluación y Ajuste del Modelo

Objetivo: Evaluar el rendimiento del modelo y realizar ajustes para mejorar su precisión.

1. **Evaluación del Modelo:**
```python
loss, accuracy = model.evaluate(X_val, y_val)
print(f'Validation Accuracy: {accuracy}')
```

2. **Ajuste de Hiperparámetros:**
   - Exploración de diferentes configuraciones de la red y parámetros de entrenamiento.
```python
from tensorflow.keras.optimizers import Adam

def build_model(learning_rate):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
        MaxPooling2D((2, 2)),
```

```
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
    return model

model = build_model(learning_rate=0.001)
model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
```

Co-creación y satélite (lección 11) Presentación, Retroalimentación y Networking del Proyecto

Objetivos:
- Preparar y presentar los resultados del proyecto.
- Discutir los hallazgos y recibir retroalimentación constructiva.

Contenidos:

1. Preparación de la Presentación

Objetivo: Preparar una presentación detallada para compartir los resultados del proyecto colaborativo.

1. **Estructura de la Presentación:**
   - Introducción al problema y objetivos del proyecto.
   - Descripción del conjunto de datos y el preprocesamiento realizado.
   - Implementación del modelo y resultados obtenidos.
   - Conclusiones y recomendaciones para futuros trabajos.

2. **Materiales de Apoyo:**
   - Diapositivas con gráficos y tablas.
   - Código fuente documentado y resultados visualizados.

2. Presentación del Proyecto

Objetivo: Presentar el proyecto final y los resultados obtenidos a compañeros y facilitadores.

1. **Exposición del Proyecto:**
   - Presentación del problema, metodología y resultados.
   - Discusión de los desafíos enfrentados y cómo se resolvieron.

2. **Sesión de Preguntas y Respuestas:**
   - Abrir espacio para preguntas y retroalimentación de los compañeros y facilitadores.

3. Discusión y Retroalimentación

Objetivo: Recibir retroalimentación constructiva y discutir posibles mejoras y futuros desarrollos del proyecto.

1. **Evaluación de Resultados:**
   - Análisis crítico de los resultados obtenidos y su significado.

2. **Retroalimentación:**
   - Recepción de sugerencias y comentarios para mejorar el proyecto.

3. **Planificación de Mejoras Futuras:**
   - Identificación de áreas de mejora y planificación de pasos futuros para el proyecto.

Talleres de Presentación de Proyectos y Networking

Objetivos:
- Presentar los proyectos finales de los participantes de manera profesional.
- Recibir retroalimentación constructiva de compañeros, facilitadores y profesionales invitados.
- Establecer conexiones con otros participantes y profesionales del sector a través de actividades de networking.

Contenidos:

1. Preparación para la Presentación

Objetivo: Asegurar que los participantes estén listos para presentar sus proyectos de manera clara y efectiva.

1. **Revisión de la Presentación:**
   - **Estructura de la Presentación:**
     - Introducción: Breve resumen del proyecto y los objetivos.
     - Metodología: Descripción del enfoque y las técnicas utilizadas.
     - Resultados: Presentación de los hallazgos y resultados del proyecto.
     - Conclusiones: Reflexiones finales y posibles mejoras futuras.
   - **Materiales de Apoyo:**
     - Diapositivas con gráficos, tablas y puntos clave.
     - Código fuente documentado y resultados visualizados.

2. **Práctica de la Presentación:**
   - Ensayo de la presentación para asegurar fluidez y claridad.
   - Preparación para responder posibles preguntas del público.
   - Asegurarse de que todo el equipo esté coordinado y listo para presentar.

2. Presentación del Proyecto Final

Objetivo: Permitir que cada equipo presente su proyecto, demostrando sus logros y el impacto de su trabajo.

1. **Configuración del Taller:**
   - **Moderador:** Designar a un moderador para introducir a cada equipo y gestionar el tiempo y las preguntas.
   - **Agenda:** Establecer un cronograma con tiempos asignados para cada presentación (por ejemplo, 10 minutos por equipo más 5 minutos para preguntas).

2. **Realización de Presentaciones:**
   - **Introducción por el Moderador:** Breve introducción sobre el propósito del taller y la agenda del día.
   - **Presentaciones de Equipos:** Cada equipo presenta su proyecto siguiendo la estructura preparada.
   - **Sesión de Preguntas y Retroalimentación:** Después de cada presentación, abrir el espacio para preguntas del público y proporcionar retroalimentación constructiva.

3. **Evaluación y Retroalimentación:**
   - **Criterios de Evaluación:** Evaluar cada presentación en términos de claridad, metodología, resultados y habilidades de presentación.
   - **Comentarios Constructivos:** Proporcionar retroalimentación específica y constructiva para ayudar a los participantes a mejorar.

3. Actividades de Networking

Objetivo: Facilitar el networking entre los estudiantes, profesionales de la industria y potenciales empleadores, creando oportunidades para establecer conexiones valiosas y explorar futuras colaboraciones o oportunidades de empleo.

1. **Actividades de Networking:**
   - **Speed Networking:** Organizar sesiones rápidas donde los estudiantes pueden interactuar con diferentes profesionales en intervalos cortos de tiempo.
   - **Mesas Redondas Temáticas:** Crear mesas redondas enfocadas en temas específicos relacionados con la inteligencia artificial, donde los estudiantes pueden discutir y compartir ideas con expertos.
   - **Intercambio de Contactos:** Proporcionar un espacio donde los estudiantes y los profesionales puedan intercambiar tarjetas de visita, perfiles de LinkedIn u otros medios de contacto.

2. **Consejos para el Networking:**
   - **Presentación Personal:** Enseñar a los estudiantes cómo presentarse de manera efectiva en un entorno de networking.
   - **Elevator Pitch:** Ayudar a los estudiantes a preparar un breve discurso que resuma quiénes son, qué hacen y qué buscan.
   - **Seguimiento:** Aconsejar a los estudiantes sobre la importancia de hacer un seguimiento con las personas que conozcan durante el evento.

Proyecto (lección 12)

colocar una indicación referente a subir el proyecto final y una imagen

English Code (lección 13) Code Testing

revisar documento drive estructura curso bootcmap, pero dejarlo para el final.

English Code (lección 14) Programming Language Python

revisar documento drive estructura curso bootcmap, pero dejarlo para el final.

Zona de recarga (lección 15) Mejores Prácticas en la Construcción de Modelos

Objetivos:
- Aprender técnicas avanzadas para optimizar modelos de machine learning.
- Implementar estrategias para mejorar el rendimiento y la eficiencia de los modelos.
- Comprender la importancia de la documentación y la reproducibilidad en los proyectos de machine learning.

Contenidos:

1. Optimización del Modelo

Objetivo: Implementar técnicas avanzadas para optimizar el rendimiento del modelo de machine learning.

1. **Ajuste de Hiperparámetros:**
   - Técnicas de ajuste fino para mejorar la precisión y eficiencia del modelo.
   ```python
   from sklearn.model_selection import GridSearchCV

   param_grid = {
       'n_estimators': [50, 100, 200],
       'max_depth': [None, 10, 20, 30],
       'min_samples_split': [2, 5, 10],
       'min_samples_leaf': [1, 2, 4]
   }
   grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
   grid_search.fit(X_train, y_train)
   print(f'Best Parameters: {grid_search.best_params_}')
   best_model = grid_search.best_estimator_
```

```

2. **Ensemble Learning:**
   - Aplicación de técnicas de ensemble learning para mejorar el rendimiento del modelo.
   ```python
   from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

   rf_model = RandomForestClassifier()
   gb_model = GradientBoostingClassifier()

   rf_model.fit(X_train, y_train)
   gb_model.fit(X_train, y_train)

   rf_predictions = rf_model.predict(X_test)
   gb_predictions = gb_model.predict(X_test)

   print(f'Random Forest Accuracy: {accuracy_score(y_test, rf_predictions)}')
   print(f'Gradient Boosting Accuracy: {accuracy_score(y_test, gb_predictions)}')
   ```

3. **Regularización:**
   - Uso de técnicas de regularización para prevenir el sobreajuste y mejorar la generalización del modelo.
   ```python
   from sklearn.linear_model import Ridge, Lasso

   ridge_model = Ridge(alpha=1.0)
   lasso_model = Lasso(alpha=0.1)

   ridge_model.fit(X_train, y_train)
   lasso_model.fit(X_train, y_train)

   ridge_predictions = ridge_model.predict(X_test)
   lasso_predictions = lasso_model.predict(X_test)

   print(f'Ridge Regression Accuracy: {ridge_model.score(X_test, y_test)}')
   print(f'Lasso Regression Accuracy: {lasso_model.score(X_test, y_test)}')
   ```

2. Documentación y Reproducibilidad

Objetivo: Comprender la importancia de la documentación y la reproducibilidad en proyectos de machine learning.

1. **Documentación del Proceso:**
   - Cómo documentar cada etapa del proceso de desarrollo del modelo.
   - Uso de herramientas como Jupyter Notebooks para combinar código y documentación.
   ```markdown

```
# Documentación del Proyecto

## Introducción
Descripción del problema y objetivos del proyecto.

## Recolección y Preprocesamiento de Datos
Detalles sobre la recolección y preparación de los datos.

## Construcción y Entrenamiento del Modelo
Descripción de los algoritmos utilizados y resultados iniciales.

## Optimización del Modelo
Técnicas de optimización aplicadas y resultados obtenidos.

## Conclusiones
Reflexiones finales y posibles mejoras futuras.
```

2. **Reproducibilidad:**
   - Importancia de la reproducibilidad en proyectos de machine learning.
   - Estrategias para asegurar que los experimentos sean reproducibles.
   ```python
   import random
   import numpy as np
   import tensorflow as tf

   random.seed(42)
   np.random.seed(42)
   tf.random.set_seed(42)
   ```

3. Mejores Prácticas en el Desarrollo de Modelos

Objetivo: Conocer y aplicar mejores prácticas en el desarrollo de modelos de machine learning.

1. **Pipeline de Machine Learning:**
   - Cómo estructurar un pipeline de machine learning desde la recolección de datos hasta la implementación del modelo.
   ```python
   from sklearn.pipeline import Pipeline
   from sklearn.preprocessing import StandardScaler
   from sklearn.ensemble import RandomForestClassifier

   pipeline = Pipeline([
       ('scaler', StandardScaler()),
       ('classifier', RandomForestClassifier())
   ])
```

```
pipeline.fit(X_train, y_train)
predictions = pipeline.predict(X_test)
```

2. **Validación y Pruebas:**
   - Importancia de la validación cruzada y las pruebas en el desarrollo de modelos.
   ```python
   from sklearn.model_selection import cross_val_score

   scores = cross_val_score(pipeline, X, y, cv=5)
   print(f'Cross-Validation Scores: {scores}')
   print(f'Mean Score: {scores.mean()}')
   ```

Zona de recarga (lección 16) Gestión de Proyectos de Machine Learning

1. Fundamentos de la Gestión de Proyectos

Objetivo: Comprender los fundamentos de la gestión de proyectos y su aplicación en proyectos de machine learning.

1. **Ciclo de Vida de un Proyecto:**
   - Iniciación, planificación, ejecución, monitoreo y cierre.
   - Aplicación del ciclo de vida a proyectos de machine learning.

2. **Roles y Responsabilidades:**
   - Definición de roles en un equipo de proyectos de IA.
   - Importancia de la colaboración y la comunicación efectiva.

2. Planificación y Ejecución de Proyectos de Machine Learning

Objetivo: Aprender a planificar y ejecutar proyectos de machine learning de manera efectiva.

1. **Definición del Alcance y Objetivos:**
   - Cómo definir el alcance y los objetivos de un proyecto de IA.
   - Establecimiento de metas claras y alcanzables.
   ```markdown
   # Definición del Proyecto

   ## Alcance
   Descripción del alcance del proyecto y sus límites.

   ## Objetivos
   Establecimiento de los objetivos específicos del proyecto.
   ```

2. **Desarrollo del Cronograma del Proyecto:**
   - Cómo crear un cronograma detallado del proyecto.
   - Uso de herramientas de gestión de proyectos para planificar y monitorear el progreso.
   ```python
   import pandas as pd

   # Ejemplo de cronograma del proyecto
   tasks = {
       'Tarea': ['Recolección de Datos', 'Preprocesamiento', 'Entrenamiento del Modelo', 'Evaluación', 'Optimización'],
       'Duración (días)': [5, 10, 15, 5, 10],
       'Dependencias': [None, 'Recolección de Datos', 'Preprocesamiento', 'Entrenamiento del Modelo', 'Evaluación']
   }

   df_tasks = pd.DataFrame(tasks)
   print(df_tasks)
   ```

3. **Asignación de Recursos:**
   - Cómo asignar recursos humanos y técnicos de manera eficiente.
   - Importancia de la asignación adecuada de tareas y responsabilidades.
   ```markdown
   # Asignación de Recursos

   ## Equipo
   Descripción del equipo de trabajo y sus roles.

   ## Recursos Técnicos
   Lista de recursos técnicos necesarios para el proyecto.
   ```

3. Metodologías Ágiles en Proyectos de IA

Objetivo: Implementar metodologías ágiles en la gestión de proyectos de machine learning.

1. **Introducción a las Metodologías Ágiles:**
   - Principios y prácticas de las metodologías ágiles.
   - Beneficios de utilizar metodologías ágiles en proyectos de IA.

2. **Scrum en Proyectos de Machine Learning:**
   - Cómo aplicar Scrum en la gestión de proyectos de machine learning.
   - Roles en Scrum: Product Owner, Scrum Master y Equipo de Desarrollo.
   - Eventos de Scrum: Sprints, Daily Stand-ups, Sprint Reviews y Retrospectives.
   ```markdown
   # Aplicación de Scrum
   ```

Roles
- Product Owner: Responsable de la visión del proyecto y priorización de tareas.
- Scrum Master: Facilita el proceso de Scrum y elimina obstáculos.
- Equipo de Desarrollo: Desarrolla y entrega el producto.

Eventos
- Sprints: Iteraciones cortas y enfocadas en la entrega de valor.
- Daily Stand-ups: Reuniones diarias para revisar el progreso y ajustar el plan.
- Sprint Reviews: Revisión del trabajo completado al final de cada Sprint.
- Retrospectives: Reflexión sobre el proceso y mejora continua.
```

3. **

Kanban en Proyectos de Machine Learning:**
- Uso de Kanban para visualizar y gestionar el flujo de trabajo.
- Principios de Kanban: Visualización del trabajo, gestión del flujo y mejora continua.
```markdown
# Aplicación de Kanban

## Tablero Kanban
- To Do: Tareas pendientes.
- In Progress: Tareas en progreso.
- Done: Tareas completadas.

## Principios de Kanban
- Visualización: Utilizar un tablero para visualizar el flujo de trabajo.
- Gestión del Flujo: Monitorear y gestionar el flujo de trabajo para evitar cuellos de botella.
- Mejora Continua: Revisar y mejorar continuamente el proceso.
```
```