Documentación: Bebedero Inteligente Impulsado por IA

Fecha de Actualización: 8 de diciembre de 2024
Versión: 1.1

1. Introducción

Este proyecto, "Bebedero Inteligente Impulsado por IA", tiene como objetivo optimizar el consumo de agua en ganado vacuno mediante un modelo de aprendizaje automático. El sistema predice el consumo de agua basándose en factores como la temperatura ambiente, el caudal del bebedero y el tiempo. Esta predicción permite ajustar el suministro de agua, evitando el desperdicio y asegurando la hidratación adecuada del ganado.

2. Arquitectura del Sistema

El sistema se compone de los siguientes elementos:

- **Sensores:** Recolectan datos de temperatura, caudal del bebedero y tiempo.
- **Modelo de IA:** Predice el consumo de agua basándose en los datos de los sensores.
- **Sistema de control:** Ajusta el suministro de agua según la predicción del modelo. (Esta parte no está implementada en el código proporcionado, es una extensión futura del proyecto.)

3. Implementación del Modelo de IA

El modelo de IA se ha implementado en Python utilizando la biblioteca scikit-learn. El código fuente se encuentra en el archivo bebedero_ia.py.

3.1. Dependencias

El código requiere las siguientes bibliotecas:

- pandas: Para la manipulación y análisis de datos.
- scikit-learn: Para el aprendizaje automático.
- joblib: Para guardar y cargar el modelo entrenado.

Estas bibliotecas pueden instalarse mediante el comando: pip install pandas scikit-learn joblib

3.2. Datos de Entrenamiento

El modelo se entrena con un conjunto de datos históricos que se encuentran en el archivo datos.csv. Este archivo debe contener las siguientes columnas:

- temperatura: Temperatura ambiente en grados Celsius.
- caudal: Caudal del bebedero en litros por minuto (o unidad equivalente).
- tiempo: Tiempo en minutos (o unidad equivalente, podría ser hora del día, etc.). Define la duración de la medición del consumo.

- consumo_agua: Cantidad de agua consumida por el ganado en litros (o unidad equivalente) durante el tiempo especificado.

3.3. Preprocesamiento de Datos

El código realiza un preprocesamiento básico de los datos, eliminando las filas con valores faltantes mediante datos.dropna(). Se recomienda realizar un análisis exploratorio de datos y considerar técnicas adicionales de preprocesamiento como la normalización o estandarización, dependiendo de las características de los datos.

3.4. Entrenamiento del Modelo

Se utiliza un modelo de Regresión Lineal (LinearRegression) para predecir el consumo de agua. El conjunto de datos se divide en conjuntos de entrenamiento (80%) y prueba (20%) utilizando train_test_split. El modelo se entrena con el conjunto de entrenamiento utilizando el método fit().

3.5. Evaluación del Modelo

El rendimiento del modelo se evalúa utilizando el Error Cuadrático Medio (MSE - Mean Squared Error) calculado sobre el conjunto de prueba. Se pueden utilizar otras métricas de evaluación como el coeficiente de determinación (R-squared) para una evaluación más completa.

3.6. Predicción

Una vez entrenado, el modelo puede utilizarse para predecir el consumo de agua con nuevos datos. El código proporciona un ejemplo de cómo realizar una predicción.

3.7. Guardado del Modelo

El modelo entrenado se guarda en un archivo llamado modelo_bebedero_inteligente.pkl utilizando la biblioteca joblib. Esto permite cargar el modelo posteriormente sin necesidad de reentrenarlo.

4. Ejecución del Código

1. Instalar las bibliotecas: pip install pandas scikit-learn joblib
2. Crear el archivo datos.csv con los datos de entrenamiento.
3. Guardar el código proporcionado como un archivo Python.
4. Ejecutar el script desde la terminal: python IA_model_v1.py

5. Futuras Mejoras

- **Integración con el sistema de control:** Implementar la lógica para controlar el suministro de agua basándose en las predicciones del modelo.
- **Exploración de otros modelos:** Evaluar el rendimiento de otros modelos de aprendizaje automático, como Random Forest o Support Vector Regression.
- **Optimización de hiperparámetros:** Ajustar los hiperparámetros del modelo para mejorar su rendimiento.

- **Interfaz de usuario:** Desarrollar una interfaz de usuario para visualizar los datos y las predicciones.
- **Implementación en un sistema embebido:** Implementar el modelo en un dispositivo de hardware para su funcionamiento en tiempo real.

Código hasta esta versión

IA_model_v1.py

C: > Users > aleja > Music > Desarrollo Web > Personal > BebederoInteligente > IA_model_v1.py

```python
# 5. Evaluar el modelo
y_pred = modelo.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Error cuadrático medio: {mse}")

nueva_temperatura = 25
nuevo_caudal = 10
nuevo_tiempo = 30

nuevos_datos = pd.DataFrame({'temperatura': [nueva_temperatura], 'caudal': [nuevo_caudal], 'tiempo': [nuevo_tiempo]})

prediccion_consumo = modelo.predict(nuevos_datos)
print(f"Predicción de consumo para temperatura={nueva_temperatura}, caudal={nuevo_caudal}, tiempo={nuevo_tiempo}: {prediccion_consumo[0]}")



import joblib
nombre_archivo_modelo = "modelo_bebedero_inteligente.pkl"
joblib.dump(modelo, nombre_archivo_modelo)
print(f"Modelo guardado como '{nombre_archivo_modelo}'")

```

Implementación del modelo con Firebase y el modelo mediante Python

IA_model_v1.py 6 | modelo_prueba1.py 4

C: > Users > aleja > Music > Desarrollo Web > Personal > BebederoInteligente > modelo_prueba1.py > ...

```python
import pandas as pd
import joblib
import firebase_admin
from firebase_admin import credentials, firestore

# 1. Cargar el modelo entrenado
try:
    modelo = joblib.load("modelo_bebedero_inteligente.pkl")
except FileNotFoundError:
    print("Error: No se encontró el archivo del modelo. Asegúrate de que 'modelo_bebedero_inteligente.pkl' esté en el mismo directorio.")
    exit()

# 2. Inicializar Firebase
cred = credentials.Certificate("ruta/a/tu/archivo/credenciales/firebase.json")
firebase_admin.initialize_app(cred)
db = firestore.client()


# 3. Obtener datos de Firebase (asumiendo una colección llamada 'mediciones')
mediciones_ref = db.collection('mediciones')
docs = mediciones_ref.get()  # Obtiene todos los documentos de la colección

datos = []
for doc in docs:
    datos.append(doc.to_dict())

df = pd.DataFrame(datos)  # Crea un DataFrame de Pandas con los datos de Firebase



if df.empty:
    print("Error: No se encontraron datos en la colección 'mediciones' de Firebase.")
```