Keras & TensorFlow

Ambos se utilizan en modelos de aprendizaje automático y redes neuronales, están estrechamente relacionados:

TensorFlow	**Keras**
Desarrollado por Google. Código abierto. Se usa para cálculos numéricos y aprendizaje automático. Proporciona una plataforma completa para la construcción y entrenamiento de modelos de machine learning y deep learning.	Desarrollador por Chollet. API de alto nivel. Se usa para construcción y entrenamiento de redes neuronales para experimentación con deep learning de manera rápida y sencilla.

Instalar TensorFlow y Keras

python -m pip install tensorflow

Integración de Keras en Tensor FLOW para ML

Desde tensor flow 2.0 keras está integrado por ello tu puedes aprovechar la simplicidad de keras y la robustez de TensorFlow. te dejo un ejemplo para montar una red neuronal:

Pasos para realizar un machine learning:

1. Definición del problema.
2. Recolección y limpieza de datos.
3. Preprocesamiento de datos.
4. Construcción del modelo.
5. Entrenamiento del modelo.
6. Evaluación del modelo.
7. Predicción.

Predicción de la probabilidad de incautaciones en ciertos municipios de colombia en fechas futuras

Definición del problema

Vamos a construir un modelo de ML con Python TensorFlow y Keras para predecir cuál será la incautación de marihuana en una fecha futura para un municipio de Colombia.

Recolección y limpieza de datos

Se adjunta el dataset:
https://www.datos.gov.co/Seguridad-y-Defensa/INCAUTACIONES-DE-MARIHUANA/g228-vp9d/about_data

El dataset contiene las siguientes columnas:

Nombre	**tipo**	**Descripción**
FECHA HECHO	DD/MM/YYYY	Es la fecha en la que ocurre la incautación. Advertencia: que nosotros queramos saber un comportamiento específico en un determinado día es algo que estadísticamente es muy difícil de calcular.
COD_DEPTO	int	Es el código de 2 dígitos del departamento de colombia. Debe ser eliminada ya que el cálculo va a ser hecho solo por municipio.
DEPARTAMENTO	str	Es el nombre del departamento donde ocurrió la incautación Debe ser eliminada ya que el cálculo va a ser hecho solo por municipio.
COD_MUNI	str	Es el código de 5 dígitos (algunos empiezan por cero)
MUNICIPIO	str	Es el nombre del municipio donde ocurrió el hecho.

		Debe ser eliminada porque es más cómodo trabajar con el código del departamento.
CANTIDAD	float	es un flotante que describe la cantidad incautada
UNIDAD	str	Es un texto que nos da la unidad de medida: KILOGRAMO como solo tiene un valor el dataset “KILOGRAMO” va a ser eliminada.

Preprocesamiento de datos

Nuestro objetivo es predecir la cantidad de marihuana incautada en una fecha futura para un municipio específico de Colombia, ahora vamos a asegurarnos de que nuestras características (FECHA HECHO, COD_MUNI, CANTIDAD) estén correctamente formateadas y listas para el ML.

1. Convertir fechas a características temporales:
   Extraer año, mes y día de la columna FECHA HECHO.

```
# Step 03 Preprocesing data
# Dates preprocesing
df['YEAR'] = df['FECHA HECHO'].dt.year
df['MONTH'] = df['FECHA HECHO'].dt.month
df = df.drop(columns=['FECHA HECHO'])
```

2. Codificación de variables categóricas:
   Asegurarnos de que COD_MUNI esté en un formato numérico también por estadísticas vamos a averiguar cuantos datos son válidos.

```
_countInputData = 0
for index, row in df.iterrows():
    try:
        int(row['COD_MUNI'])
        _countInputData = _countInputData + 1
    except ValueError:
        del_codes_by_int_error.append(index)
df = df.drop(del_codes_by_int_error)

# Convert place code to INT
df['COD_MUNI'] = df['COD_MUNI'].astype(int)
```

   Hay que recordar que los datos son introducidos por humanos y a veces hay errores

en la introducción como poner una letra o un guión.

3. Seleccionar las características y el target

```
# Select target X and result Y
X = df[['YEAR', 'MONTH', 'COD_MUNI']]
y = df['CANTIDAD']
```

4. Dividir el dataset entre entrenamiento y prueba:

```
# dataset >> fit & test
_test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=_test_size, random_state=42)
```

5. Normalización:
   Escalar las categorías numéricas para que todas estén en el mismo rango.

```
# Normalized
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

6. Separación del target:
   Separa la característica objetivo CANTIDAD del resto de características. (ya se hizo en el paso 3)

Construcción del modelo

El proceso con el cual nosotros estamos trabajando es un problema de regresión.

Para nuestro caso el modelo secuencial de queras es la forma más simple de construir redes neuronales. Como saber cuantas capas y neuronas utilizar:

Hay que recordar que esto más que una ciencia exacta es un arte, a medida que nos enfrentemos a problemas vamos a ganar un cálculo.

- Comenzar simple: iniciar con un modelo simple y agregar complejidad según sea necesario.
- Si usas menos capas es por que el problema es simple y si usas más capas es por que el problema es complejo no lineal.
- Número de neuronas por capa: se suele comenzar con un número mayor de neuronas (128 o 64) y luego se va reduciendo (32 o 16).

```
# Step 04 create model
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))
```

Advertencia: recuerden que esto es más un arte que una ciencia… hay que probar otras arquitecturas… y la manera de probar es cambiando el modelo y los hiperparametros.

Pegar aquí el dibujo …

Entrenamiento del modelo

En este paso el modelo va a aprender a predecir la cantidad de marihuana incautada basada en nuestros datos. Antes de entrenar necesitamos hacer lo siguiente:

- Compilar el modelo.
    - Definir el optimizador (Adam) .
    - Definir función de pérdida.
    - Definir las métricas.

```
# Step 05 fit the model
# Compile model
_learning_rate=0.001
optimizer = Adam(learning_rate=_learning_rate)
model.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['mean_squared_error'])
```

- Entrenamiento del modelo:
    - Definir un número de épocas.
    - Definir cuánta paciencia va a tener el modelo.
    - Agregar una función de callback por si el entrenamiento no sirve se detenga.
    - Utilizamos los datos de entrenamiento X_train y_train.
    - Definir el tamaño del lote.
    - Usaremos los datos de validación de datos del modelo.

```
# fit
# FIT with early stopping
_epochs = 100
_patience = 10
early_stopping = EarlyStopping(monitor='val_loss', patience=_patience, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=_epochs, validation_data=(X_test, y_test), callbacks=[early_stopping])
_epochs_trained = len(history.epoch)

model.fit(X_train, y_train, epochs=_epochs, validation_data=(X_test, y_test), callbacks=[early_stopping])
```

Advertencia: el proceso puede tardar muchísimo tiempo… todo depende de la máquina.

Evaluación del modelo

Utilizamos los datos X_test y_test para medir el rendimiento del modelo y asegurarnos de que no está sobre ajustado o los datos de entrenamiento.

Como nuestro modelo es un problema de regresión los parámetros (Accuracy, Precision, Recall, F1-Score, ROC, y AUC) no son válidos debido a que estos problemas son solo para clasificación. Para los problemas de regresión se usan las métricas: **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)**, **R-squared (R^2)**.

```
# Step 06 Evaluate
loss, mse = model.evaluate(X_test, y_test)
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

Predicción

```python
# Step 07 Predit
# INPUT DATA TO PREDICT:
_YYYY = 2024
_MM = 11
_COD_MUN = 5001 # 5001 MEDELLIN
input_data = {
    'YEAR': [_YYYY],  # Año de la predicción
    'MONTH': [_MM],  # Mes de la predicción
    'COD_MUNI': [_COD_MUN]  # Código del municipio
}

input_df = pd.DataFrame(input_data)
input_scaled = scaler.transform(input_df)

# Predit
prediction = model.predict(input_scaled)
```

Ejecución del script a prueba y error

Para el modelo 64,16,1 a 20 epocas obtenemos los siguientes datos:

```
TEST RESULTS:
LOSS & Mean Squared Error on test set: 9309.2958984375
Mean Absolute Error on test set: 10.209970328122528
R-squared on test set: 0.027279053638858 52
```

LOSS:

El MSE es una medida del promedio de errores cuadrados entre los valores predichos por el modelo y los valores reales.

MAE:

Mide el promedio de los errores absolutos entre los valores predichos y los valores reales, el número indica cuántas unidades está desfasada nuestro modelo. Lo que se busca es un valor muy bajo.

R2:

Es la medida de varianza entre la variable independiente (cantidad de marihuana incautada) que es explicada por nuestro modelo. un valor de 0.0272 significa que nuestro modelo es solo 2.72% efectivo para predecir.

Nuestro modelo no sirve…

Ahora vamos a empezar a tabular para ver donde se ajusta mejor nuestro modelo:

LR: tasa de aprendizaje.
Épocas: cuantas veces relee los datos de entrada para entrenarse.
LOSS: qué tan lejos estamos de la solución.
MSE: diferencia con respecto al valor real.
R2: porcentaje de verdad del modelo.

Arquitectura	**LR**	**Épocas**	**LOSS**	**MSE**	**R2**