Primera hoja de notas:
Github del profesor: pablotalavante

wind_ava[energy]

wind_ava.head():
- data type: día
- energía

wind_cop.head()
falta la energía --> predecir

cuando vayamos a construir el data frame final --> eliminar resto de localizaciones
Cuando vemos una tabla con los datos: p54.162.1, hay que mirar la última hoja donde pone qué es cada cosa. En este caso es "Vertical integral of temperature" del punto 1, eliminar todos los que no sean .13

1. Fase EDA: describir dataset (ya están bastante limpios)
    - quitar todo lo que no son .13. Primero quiero todas las columnas: **for c in wind_ava.columns: print(c) if not c.endswith('.13'): wind_ava.drop(c, axis = 1)** El taxi es para que haga las columnas y no las filas. Se añade inplace = True para que se reemplace y así se guarden los datos que hemos guardado.
      Otro problema que se crea es que se borra la columna de energía y de datatime. Para la de energy añadiremos también en el primer **"and c!= 'energy'"**. El datatime lo necesitamos pasar a índice para que se pueda utilizar en el eje de las x en las gráficas. Para ello **"wind_ava = wind_ava.set_index('datatime')"**.

    - Ahora vamos a ver qué hacer cuando faltan valores. Un valor faltante no es un 0, y un 0 no es un valor faltante. **"wind_ava.isna()"** te indica si hay algún valor faltante. Si hacemos **"wind_ava.isna().sum"** para que así nos indique en cada columna cuales son los valores faltantes. En este caso, no hay ninguno.

    - Ahora hay que comprobar si hay columnas constantes **"wind_ava.std()"**. Para ello, vamos a mirar la desviación de cada columna. Si alguna es igual a 0, es que es constante.

Si utilizamos .plot(), nos va a sacar las gráficas. Si dentro del paréntesis ponemos plot(rot= 45) para que gire las variables en el eje de las x y que así se puedan leer mejor.

Otra de las cosas que podemos utilizar es el .plot.box(), sirve para los valores atípicos. Aun así no todos los valores atípicos son malos por así decirlo. Pueden que hayan producido estos valores atípicos.

Otra cosa que podemos utilizar es el **".plot.hist(bin=45)"**. Esto lo que hace es hacer un gráfico con las frecuencias en las que aparece cada dato. Lo del bin= 45 es para ver en

cuántas columnas se divide el gráfico. Además, podemos hacer **“wind_ava[‘log_energy’] = np.log(wind_ava[‘energy’])”** y luego lo de wind_ava[
2. Validación cruzada:

**Exportar modelo
model.fit() //Entrenamos
creamos el objeto .pkl (Ver p1)

Regresión

INNER:
inner lo que hace es coger todo el rectángulo del train, y ahí dividirlo en segmentos, en los cuales luego hace los K-fold

OUTER:
Ahora con esto, lo que vamos a hacer es ver cuanto de bueno es para el futuro. Para ello, vamos a comprobar cada uno de los modelos del train con nuevos datos, los cuales son los de test, con los distintos K-Fold del inner, y nos saca una métrica de cuánto de bien van esos hiperparámetros. Si se hacen varios outers, es porque también se harían k-fold dentro de los datos de test.
Después, con estos hiperparamétros elegidos, ya se utilizan para todos los datos del modelo, pudiendo así darse una configuración de hiperparámetros mejor de la que habíamos elegido antes.

Punto 2

- Establecer los kfolds / método (KNN…) / hyperparámtros?
- **Time Series Split** = Outer
- Inner tambien?

Punto 3

- Hacer 3 modelos?

uc3m | Universidad Carlos III de Madrid

Doble grado en Ingeniería Informática y Administración y Dirección de Empresas

Aprendizaje Automático

Práctica 1

Grupo 801

Jaime Ballesteros - 100454114
Nicolás Lamotte - 100454275

ÍNDICE

1.Realización de un EDA

Para empezar con este trabajo lo primero que se va a realizar es un EDA (Análisis Exploratorio de Datos). Los datos que se analizan son los wind_ava, los cuales son 22 datos meteorológicos que a su vez tienen una cuadrícula de 5x5, proporcionando esos 22 datos de 25 localizaciones. Además de estos 22 datos meteorológicos, hay dos datos más, los cuales son el data_time y el de

De estas 25 localizaciones, solo interesan la localización 13, la referente a Sotavento. Debido a esto, en nuestros datos eliminaremos

** Verificar datos atípicos: Nuestro caso tiene una distribución exponencial, por lo tanto, no es interesante hacerlo (ver box plot). Quitarlos sería no poder predecir este tipo de valores. valor numerico no clasificatorio

**Analizando fechas: es un string, pero accediendo de modo date_time, es más fácil para analizarlo y poder acceder a cada uno de ellos
pd.to_datetime → objeto del tipo datetime
x.year
x.month
hay huecos?
hay más frecuentes?
…

2. Outer/Inner Evaluation

Inner Evaluation (con train)

- Time Series Split, para ello se tienen los datos y se divide entre train y test (siendo el test la parte del outer, se pueden dividir por años) Este train se va a dividir en K-folds, también pueden ser por años las divisiones. Si tengo 10 años, en K=1, hace train el primer año y evalúa con el segundo, dejando el resto sin nada para no afectar al error; en K=2, hace el train con el año 1 y 2, y evaluar con el 3, y así todos tus K.
  **Comprobar que se pillen los datos ordenados temporalmente para ada split
  Rolling WIndow → construir features con las ya existentes (ABG 30/7 days)
  Se puede crear la features de energía producida ayer
- KNN
- GridSearch
- 3 Folds

Outer → con test