Unidad	5
Entrega	Documento en Jupyter Netbook

1. Enunciado

1. Utiliza el siguiente enlace para descargar el Diabetes Dataset, junto con su fichero de nombres. Utiliza un Jupyter Notebook y el paquete Pandas para abrir el archivo y presentarlo en formato DataFrame, donde el nombre de las columnas debe corresponder con el nombre real de las variables.

   <https://www.kaggle.com/uciml/pima-indians-diabetes-database>

   a. Utiliza el método info de Pandas sobre el DataFrame para comprobar que todas las variables del problema son numéricas. ¿Cuántos valores faltantes tiene cada variable? Explica el nombre y el significado de las variables que aparecen en el dataset. ¿Tiene sentido que alguien tenga un índice de masa corporal o una presión sanguínea igual a cero? En las variables que corresponda, sustituye los valores que no tengan sentido por valores np.nan. Vuelve a utilizar el método info() de Pandas. ¿Cuántos valores faltantes aprecias ahora?

   b. La mayoría de los métodos de Machine Learning no pueden tomar como datos de entrada datasets en los que haya valores faltantes. Para poder usar estos datasets se pueden tomar dos enfoques. Uno de ellos es eliminar todos los registros en los que haya valores faltantes, de forma que nos quedamos con un nuevo dataset de menor tamaño que el original. Este enfoque es rápido y sencillo, pero tiene inconvenientes. Si el número de instancias con valores faltantes es grande, podemos quedarnos con un dataset demasiado pequeño. Además, es posible que existan razones o correlaciones por los cuales los datos faltan, por lo que al eliminarlos estaríamos eliminando del dataset un tipo particular de instancia que no se tendría en cuenta al entrenar el modelo, socavando su capacidad de predicción. Podemos comprobar esta circunstancia haciendo algunos análisis estadísticos de los datos presentes y los datos faltantes.

      Usando el DataFrame en el que los valores faltantes han sido marcados como np.nan, dibuja en una misma gráfica dos histogramas, ambos con nivel de transparencia (alpha) igual a 0.7, y de forma que ambos sean de colores distintos. Uno de los histogramas corresponderá al histograma de la variable “DiabetesPedrigreeFunction” de las instancias en las que “Insulin” tiene un valor numérico. El otro histograma corresponderá al histograma de la variable “DiabetesPedrigreeFunction” de las instancias en las que el valor de “Insulin” es un valor faltante. ¿Ambos histogramas son iguales? ¿Podemos pensar que los valores de “Insulin” son faltantes con independencia de cuál sea el valor de “DiabetesPedrigreeFunction”?

   c. Otro de los enfoques que se pueden utilizar es el de imputar o rellenar los valores faltantes del dataset. No existe una técnica general que nos permita encontrar la mejor imputación de valores faltantes, pero existen

algunos enfoques básicos que, por su sencillez, pueden ser las primeras ideas a aplicar cuando trabajamos con un dataset desconocido.

Permitiremos la presencia de valores faltantes tanto en el conjunto de entrenamiento como en el de test. Sin embargo, hay un factor imprescindible a tener en cuenta: debemos imputar los valores faltantes en el conjunto de test de acuerdo a las reglas y parámetros establecidos en el conjunto de entrenamiento. De lo contrario, estaríamos falseando la capacidad predictora de nuestro modelo debido a un fenómeno conocido como data leakage, en el que la predicción sobre el conjunto de test se evalúa utilizando una información sobre el conjunto de la que no se va a disponer cuando se aplique el modelo a instancias que provengan del mundo real. Para evitar el data leakage utilizaremos la clase Pipeline de Scikit Learn en la que se definirán dos pasos: primero el método de imputación y después el modelo de Machine Learning con el que se resolverá el problema.

En este apartado vamos a imputar todos los valores faltantes del dataset utilizando un valor constante igual a cero. Declara un objeto de la clase SimpleImputer de Scikit Learn en el que el argumento “strategy” sea igual a “constant”. Declara un objeto de la clase RandomForestClassifier de Scikit Learn con los valores por defecto. Declara un objeto de la clase Pipeline de Scikit Learn en el que en la variable “steps” se utilice primero el objeto de SimpleImputer y después el objeto de RandomForestClassifier. Declara un objeto de la clase StratifiedKFold en el que la variable ‘n_splits’ sea igual a 10 y la variable “random_state” sea igual a 42. Utiliza el método “cross_val_score” de Scikit Learn para obtener 10 valores de la métrica del clasificador aplicado a este conjunto de datos. En los argumentos del método debemos utilizar el objeto de la clase Pipeline que hemos definido anteriormente, un DataFrame X donde se han tenido en cuenta solo las variables predictoras y un array de numpy Y en el que aparezcan los valores de la target. El argumento “scoring” debe ser igual a “accuracy", y el argumento ‘cv’ debe ser igual al objeto de la clase “StratifiedKFold” que hemos declarado antes. Guarda los valores de las métricas obtenidas en un array llamado accuracy_constant.

d. En este apartado vamos a imputar todos los valores faltantes del dataset utilizando el valor de la media de la variable en el conjunto de entrenamiento. Repite los pasos del apartado anterior pero de forma que el argumento “strategy” del objeto de la clase SimpleImputer sea igual a “mean”. Guarda los valores de las métricas obtenidas en un array llamado accuracy_mean.

e. En este apartado vamos a imputar todos los valores faltantes del dataset utilizando el valor más frecuente de la variable en el conjunto de entrenamiento. Repite los pasos del apartado anterior pero de forma que el argumento “strategy” del objeto de la clase SimpleImputer sea igual a “most_frequent”. Guarda los valores de las métricas obtenidas en un array llamado accuracy_most_frequent.

f. En este apartado vamos a imputar todos los valores faltantes del dataset utilizando un modelo que permita asignar a cada valor faltante el valor numérico que estime más adecuado. Utilizamos un imputador de tipo KNN, en el que el valor numérico de un valor faltante se decide mirando cuál es el valor de sus k vecinos más próximos. Un parámetro fundamental del modelo será el valor de k, que podremos variar para intentar mejorar la capacidad predictiva del pipeline en su conjunto.

   Repite los pasos del apartado anterior pero de forma que en vez de declarar un objeto de la clase SimpleImputer, declares un objeto de la clase KNNImputer de Scikit Learn. Donde el valor de la variable "n_neighbors" sea igual a 5. Declara el objeto de la clase Pipeline de forma que en la variable "steps" se utilice primero el objeto de KNNImputer y después el objeto de RandomForestClassifier. Declara el objeto de la clase StratifiedKFold y utiliza el método "cross_val_score" de la misma forma que anteriormente. Guarda los valores de las métricas obtenidas en un array llamado accuracy_knn_k5.

g. Representa en un gráfico de violín los valores de los arrays accuracy_constant, accuracy_mean, accuracy_most_frequent y accuracy_knn_k5. ¿Qué método de imputación parece más apropiado para este conjunto de datos basándonos en los valores de la accuracy obtenidos para el dataset de test?

h. Busca información sobre el fenómeno de data leakage. Haz un pequeño resumen de un párrafo de la información que hayas obtenido, especificando de dónde la has obtenido, y pon un ejemplo de caso de uso en el que se produzca data leakage por no separar convenientemente los conjuntos de entrenamiento y test.

2. Utiliza el siguiente enlace para descargar el California *Housing Prices* Dataset. Utiliza un Jupyter Notebook y el paquete Pandas para abrir el archivo y presentarlo en un DataFrame llamado df_house donde el nombre de las columnas debe corresponder con el nombre real de las variables.

   <https://www.kaggle.com/camnugent/california-housing-prices>

Elimina la variable "ocean_proximity"de df_house. Elimina las instancias que contengan valores faltantes. Crea un array de nombre y_house con los valores de la variable "median_house_value". Elimina la variable "median_house_value" del DataFrame df_house. Crea conjuntos de train y test usando train_test_split con test_size=0.3 y random_state=12.

a. Crea un ColumnTransformer que aplique StandardScaler a las variables "longitude" y "latitude" y aplique PowerTransformer (con method='box-cox') y StandardScaler al resto de variables. Crea un Pipeline que tenga como primer paso el ColumnTransformer así definido, como segundo paso un objeto SelectKBest y como tercer paso un objeto Linear Regression. Crea un objeto TransformedTargetRegressor que utilice como 'regressor' el Pipeline antes definido y como 'transformer' una transformación PowerTransformer (con method='box-cox'). Este último paso aplicará de forma adecuada una transformacion de Box-Cox sobre la variable objetivo.

b. Con el objeto TransformedTargetRegressor así definido, define las variables de entrada del método SelectKBest de forma que score_func sea igual al método f_regression y k sea igual a 4. Asigna a este TransformedTargetRegressor el nombre feat_sel_corr. Al usar el método f_regression hacemos que el estadístico mediante el cual se seleccionan las variables sea la correlación de Pearson entre las variables y la variable objetivo. Utiliza un método cross_val_score para hacer una cross validation donde el primer argumento sea feat_sel_corr, el segundo y el tercero sean X_train e y_train y el método de scoring sea 'neg_root_mean_squared_error'. ¿Qué valores de RMSE se obtienen?

c. Entrena el objeto feat_sel_corr sobre X_train, y_train. ¿Qué variables se han seleccionado para realizar la predicción sobre futuros datos?

d. Repite los pasos del apartado b, donde ahora el objeto creado con TransformedTargetRegressor se llama feat_sel_mi y donde los argumentos del método SelectKBest se definen de forma que score_func sea igual al método mutual_info_regression y k siga siendo igual a 4. Al utilizar el método mutual_info_regression hacemos que el estadístico mediante el cual se seleccionan las variables sea igual a la información mutua. Utiliza el método cross_val_score para obtener los resultados de la cross validation, como en el apartado b, donde ahora el método es feat_sel_mi. ¿Obtienes mejores resultados que en el apartado anterior? ¿Cuál crees que es el motivo?

e. Entrena el objeto feat_sel_mi sobre X_train e y_train. ¿Qué variables se han seleccionado? ¿Son las mismas que en el apartado c?

f. En esta sección utilizaremos un método de selección de features llamado Recursive Feature Elimination (RFE). Este método necesita hacer uso internamente de un modelo predictivo, al que llamaremos estimator. El modelo usado como estimator no tiene por qué ser el mismo que el modelo predictivo que usemos en nuestro Pipeline de Machine Learning, la única condición es que el modelo usado como estimator tenga un atributo que nos dé información sobre feature importances. De esta forma, podemos usar como estimator una regresión lineal o un random forest, por ejemplo, mientras que en el modelo predictivo de nuestro Pipeline podemos usar cualquier modelo como regresión lineal o random forest, o también modelos que no den información sobre feature importance, como las redes neuronales.

   Declara un objeto estimator de la clase RandomForestRegressor de Scikit Learn. Declara un objeto rfe de la clase RFE de Scikit Learn. La variable “estimator” debe ser igual al objeto estimator declarado anteriormente. La variable “n_features_to_select” debe ser igual a 4. Declara un objeto linmod de la clase LinearRegression de Scikit Learn. Declara un objeto pipeline de la clase Pipeline de Scikit Learn. La variable “steps” debe ser un array en el que se especifique un primer paso correspondiente al objeto rfe y un segundo paso correspondiente al objeto linmod. Utiliza los conjuntos de train y test definidos en el enunciado. Entrena el objeto pipeline sobre el conjunto de entrenamiento. Una vez entrenado, obtén los nombres de las cuatro variables que han sido seleccionadas como entrada del modelo predictivo. ¿Son las mismas variables que en los apartados anteriores?

g. Utiliza los TransformedTargetRegressor entrenados en los apartados c y e y el Pipeline entrenado en el partado f para hacer una predicción sobre el conjunto de test, utilizando como métrica el Root Mean Square Error. ¿Qué método de selección de variables daría mejor resultado para este conjunto de datos, considerando que hemos fijado k=4 para todos los métodos? ¿Has encontrado algún problema al utilizar el método de Recursive Feature Elimination?

2. Detalles de la entrega

- Las respuestas de la actividad se deberán entregar en un Jupyter Notebook en el que se haya respondido a cada apartado en una celda independiente, en el orden de las preguntas de este documento. El código de cada celda se debe poder ejecutar para comprobar las respuestas aportadas. Las preguntas teóricas se deben responder en una celda de formato texto.
- Subir de forma grupal el documento a la actividad en el campus virtual.