Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Tarea 1: Valores faltantes

Victor Manuel Gómez Espinosa

14 de septiembre de 2020

1. Métodos de imputación de valores faltantes

Se realizó una clase en Python llamada MissingValueImputation, que realiza la imputación de datos faltantes en un DataFrame con alguno de los siguientes 5 métodos:

1. Métodos no estadísticos: LOCF (Last observation carried forward)
2. Métodos estadísticos: mean_mode (media para atributos numéricos, moda para atributos categóricos)

Métodos de Machine Learning. Ajuste mediante RandomizedSearchCV con 3-Fold:

3. Vecinos más cercanos: knn (KNeighbors: k: [2-40])
4. Arboles de decisión: trees (XGBoost: learning rate: [0.1-0.6], n_estimators: [50-300], subsample: 0.9, regularización: gamma: [0-100], lambda: [1-10])
5. Redes neuronales: MLP (epocas: 100, optimizador: adam, función de costo: binary_crossentropy (o categorical_crossentropy), métrica: accuracy, hiddenlayers: 2 con 50 neuronas, función de activación: relu. outputlayer: función de activación: sigmoid (o softmax) con 1 (o número de clases) neuronas.)

Se tomaron 10 conjuntos de datos con valores faltantes de la base de datos de KEEL. Los 10 conjuntos de datos presentan diferentes características, la cantidad de ejemplos que esta entre 250 y 1000, la mayoría son de 2 clases, máximo 6 y de 5 a 50% de datos faltantes (Figura 1.1).

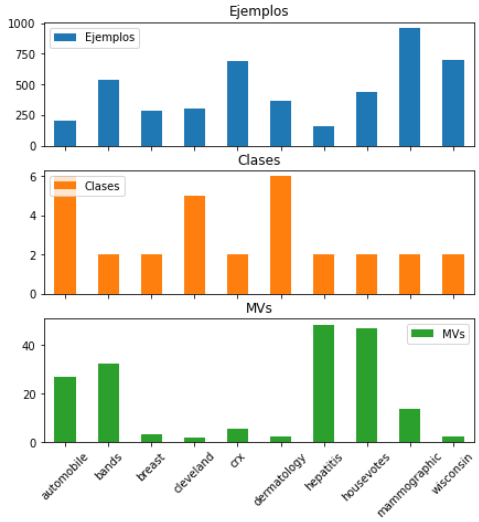

Figura 1.1: 10 Conjuntos de datos con valores faltantes (MVs).

Para generar datos para comparar cada método de imputación, para cada conjunto de datos se aplicó cada uno de los métodos de imputación y para cada uno se utilizaron 3 modelos de Machine Learning para clasificación ajustados mediante búsqueda aleatoria y validación cruzada (10x2) y empleando la métrica de accuracy para generar 30 datos por cada método de imputación. Los modelos son los siguientes:

1. KNeighbors: k: [2-40]
2. XGBoost: learning rate: [0.1-0.6], n_estimators: [50-300], subsample: 0.9, regularización: gamma: [0-100], lambda: [1-10]
3. MLP, epocas: 100, optimizador: adam, función de costo: binary_crossentropy(o categorical_crossentropy), hiddenlayers: 2 con 50 neuronas, función de activación: relu. outputlayer: función de activación: sigmoid (o softmax) con 1 (o número de clases) neuronas.

2. Análisis estadístico de los resultados

A partir de las muestras generadas como se explicó anteriormente, mediante un análisis descriptivo se pudo observar (Figura 2.1) que todos los métodos tienen una variación en un rango de 0.5 y 0.97, con mediana muy similar, pero aparentemente el método de imputación basado en arboles de decisión fue un poco mejor, ya que su rango es ligeramente más reducido.

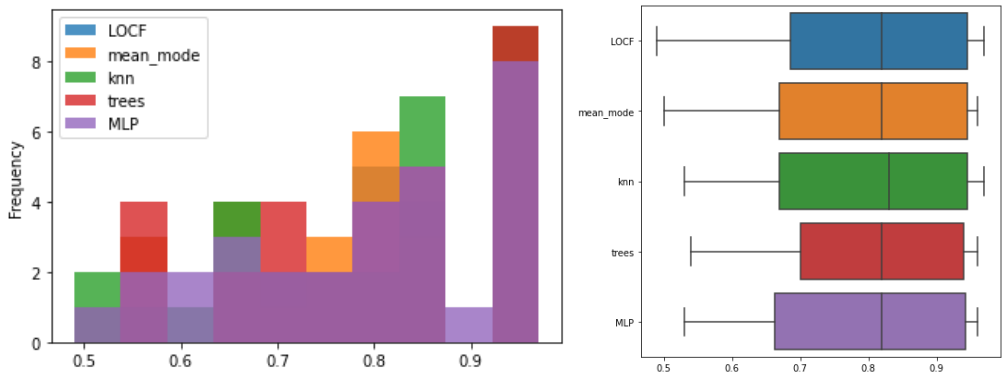

Figura 2.1: a) Histograma, b) Box-plot de los datos generados por cada método de imputación.

Posteriormente se aplicó la transformación de Box-Cox para aproximar los datos a normales univariados y corroborando con la prueba Shaphiro para poder aplicar una prueba t de diferencia de medias, las cuales para todos los casos resultó que ningún método es significativamente mejor que otro.

Por otro lado, se realizó también una comparación Bayesiana en la cual se pudo observar que el método basado en arboles fue mejor que todos los demás, seguido de los métodos de vecinos más cercanos, media o moda, LOCF y finalmente el método basado en redes neuronales (Figura 2.2).

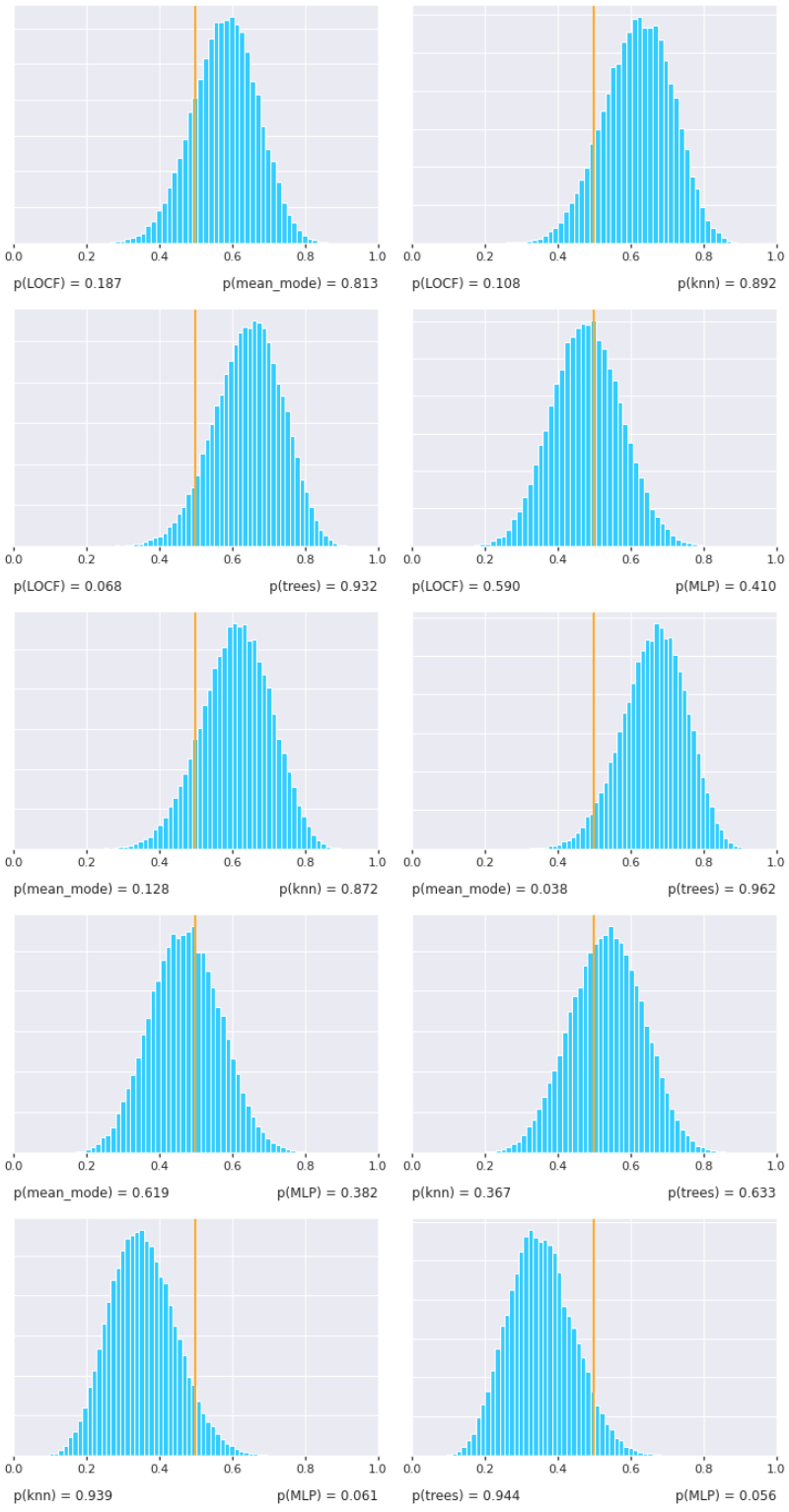

Figura 2.2: Comparación Bayesiana de los clasificadores. Mejor clasificador: trees.

Conclusiones:

- El método basado en arboles fue el mejor en este experimento, algunas de las posibles causas son que para este método se realizo un ajuste por búsqueda aleatoria y se empleó regularización, además de las ventajas mismas de la implementación XGBoost la cual no necesita que el conjunto de datos sea estandarizado, el método de construcción de los arboles aprende con los datos que decisión tomar cuando hay valores ausentes y también como otros métodos basado en árboles, estos logran captar cuales son las características más importantes en la construcción de los árboles.
- Por otro lado, el método basado en redes neuronales fue el peor, pero hay que considerar que en este caso no se empleó regularización, se utilizó un mismo modelo, es decir por tiempo computacional no se hizo búsqueda aleatoria sobre los parámetros para ajustar el mejor, esto pudo causar un ajuste pobre, además que los métodos de redes neuronales son especialmente buenos cuando se tiene una gran cantidad de datos y en este caso el tamaño máximo fue de 1000 aproximadamente lo cual es muy poco.
- Otro factor a considerar es que no se tomo en cuenta si había clases desbalanceadas, lo cual también afecta la clasificación.