Practicas 1- 4

Inteligencia de Negocio

Gustavo Sobrado Aller

UO286277

71777616K



TABLA DE CONTENIDO

[1. PRACTICA 1 2](#_Toc181667158)

[1.1 PARTE 1 2](#_Toc181667159)

[1.2 PARTE 2 7](#_Toc181667160)

[1.3 PARTE 3 13](#_Toc181667161)

[2. PRACTICA 2 18](#_Toc181667162)

[2.1 PARTE 1 18](#_Toc181667163)

[3. PRACTICA 3 25](#_Toc181667164)

[4. PRACTICA 4 28](#_Toc181667165)

[4.1 PARTE OBLIGATORIA 28](#_Toc181667166)

[4.2 PARTE OPCIONAL 1 33](#_Toc181667167)

[4.3 PARTE OPCIONAL 2 35](#_Toc181667168)

# PRACTICA 1

## PARTE 1

1. Añade una tercera imagen al gráfico con un ruido de desviación típica 64. Debes obtener el resultado que se muestra en esta transparencia.

Texto

Descripción generada automáticamente

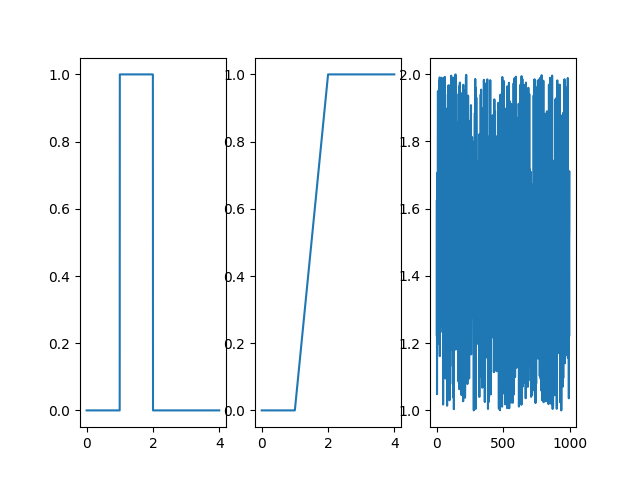
Interfaz de usuario gráfica

Descripción generada automáticamente

1. Dibuja las funciones de densidad, de distribución y 1000 muestras de una distribución de probabilidad uniforme en [2,3]. Debes obtener el resultado que se muestra en esta transparencia.

Texto

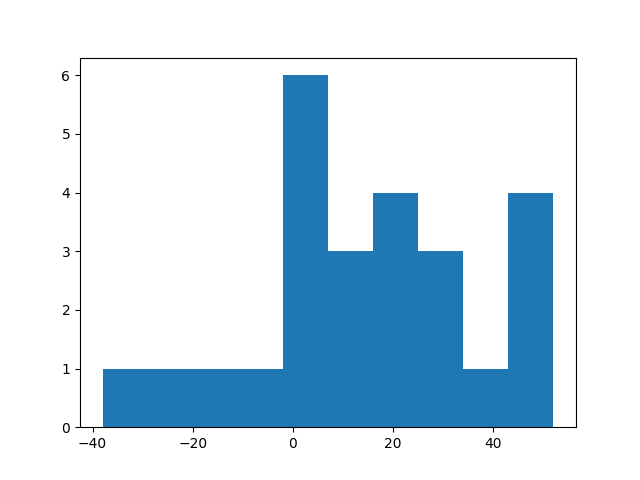
Descripción generada automáticamente



1. Usa una prueba de Wilcoxon para realizar la misma comprobación. ¿Cuál es el p-valor? ¿La conclusión es la misma, o es diferente?

Texto

Descripción generada automáticamente



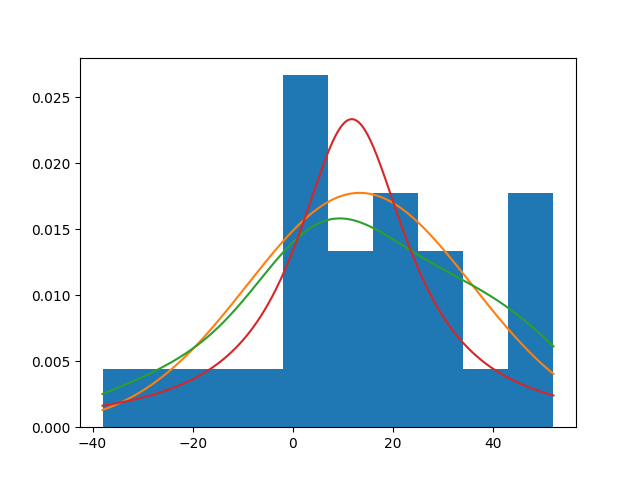
El p-valor es: 0.007531

Aunque cambiemos de teste los tiempos siguen siendo significativamente diferentes.

1. Ajusta también una distribución de Cauchy a los mismos datos. Debes obtener el resultado mostrado en la transparencia (curva verde).

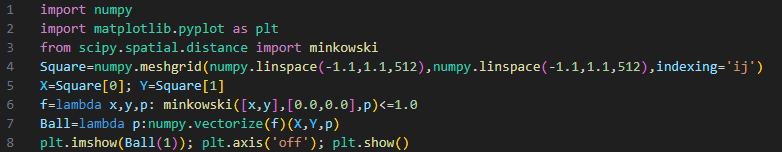
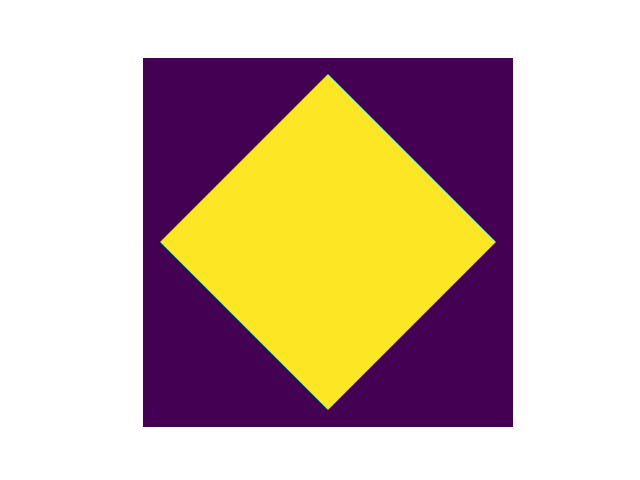
Texto

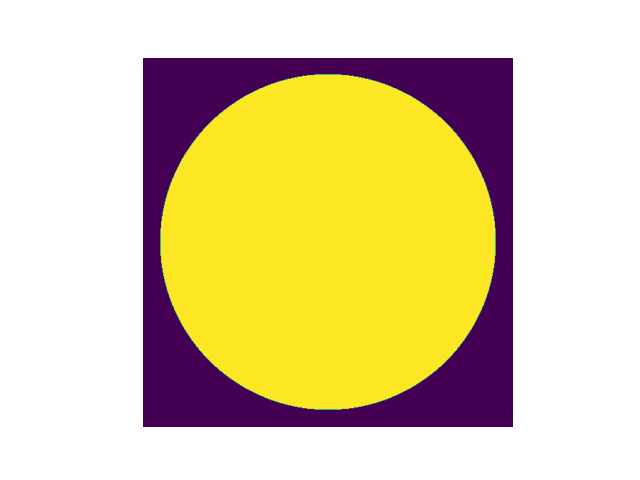
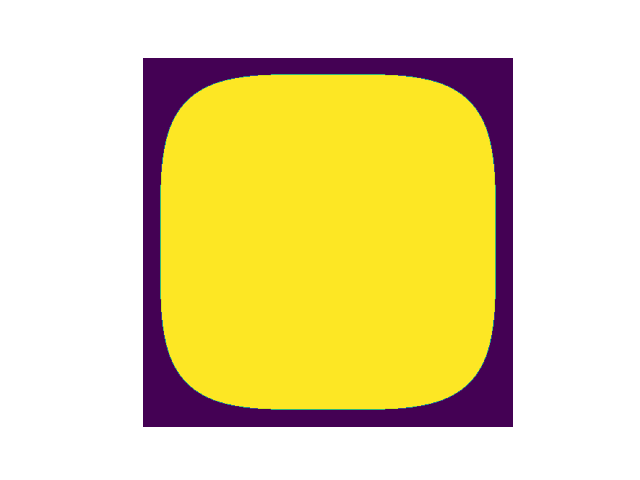
Descripción generada automáticamente



1. Dibuja la bola de tamaño 1 para las distancias de Minkowski de órdenes 1,2 y 4.

Lo unico que hay que hacer en este ejercicio es cambiar el valor de ball(x). Poniendo en x el valor 1,2,4.



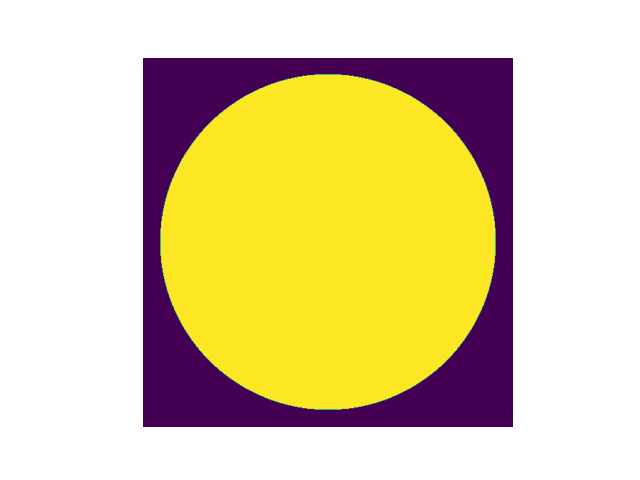
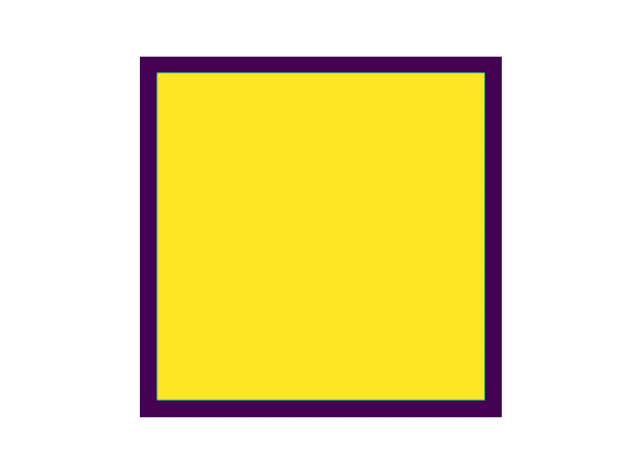
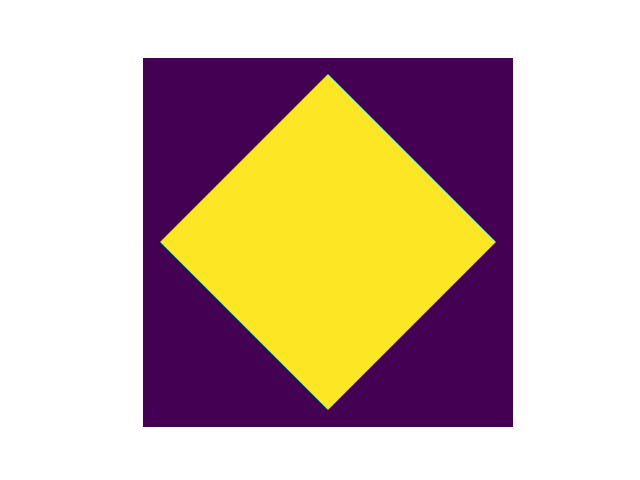


Dibuja la bola de tamaño 1 para las distancias euclídea, Chebyshev y Manhattan.

Para este ejercicio cambiamos el tipo de distancia que queremos usar, es decir en el código sustituimos en f minkowsky por euclidean, por ejemplo, y quitar la p de todo el código.

Texto

Descripción generada automáticamente



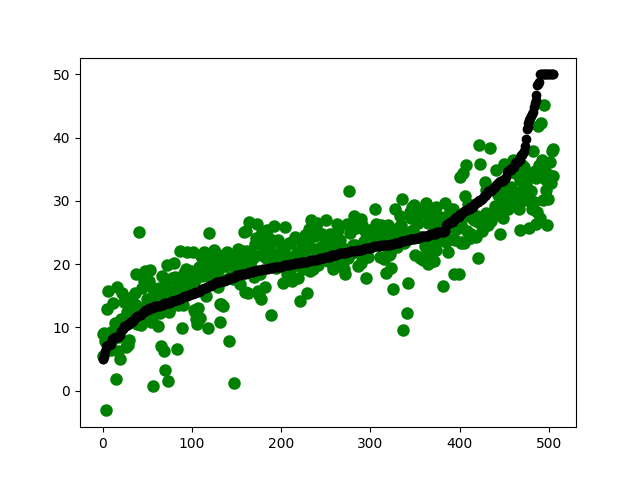
## PARTE 2

* 1. Modifica sklearn-1-mod.py para que se muestre cualquier otro modelo de regresión (polinomio, red neuronal, u otra cualquiera) y superpon las gráficas de LinearRegression, SVR y RandomForestRegressor para comparar gráficamente los ajustes.

Añadimos el modelo MLPRegressor en el archivo multivariate, por ejemplo.

Texto

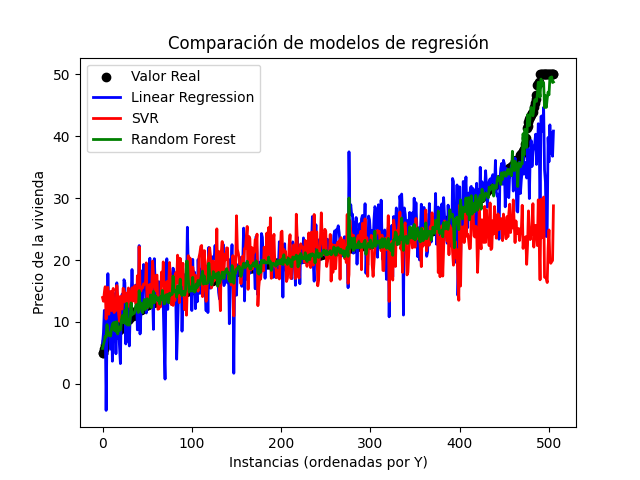
Descripción generada automáticamente



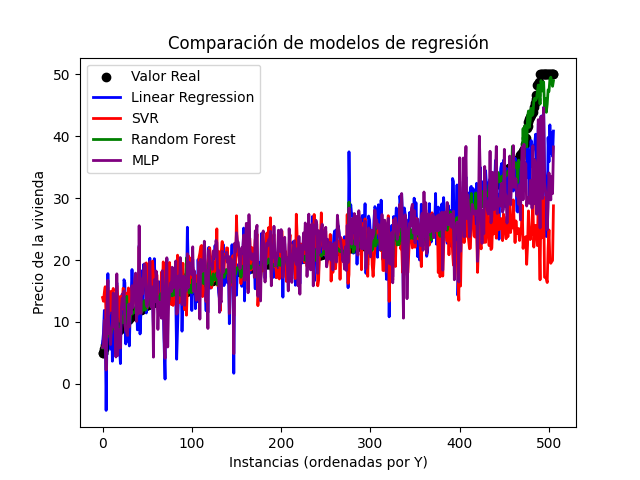
Ahora superponemos las gráficas de LinearRegression, SVR y RandomForestRegressor en el nuevo archivo sklearn-mod1.py.

Texto

Descripción generada automáticamente



Si superponemos los 4 modelos:

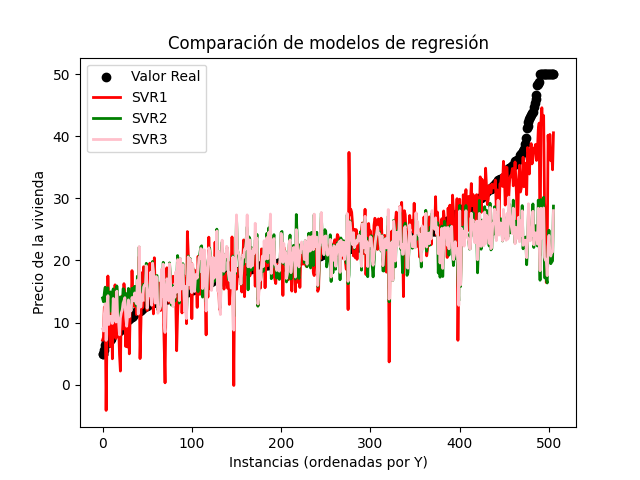


* 1. Prueba con diferentes tipos de kernel y valores de parámetro C en SVR ().

El código se encuentra en sklearn-mod2.py

Texto

Descripción generada automáticamente



* 1. Prueba con valores de k diferentes de 1 y de 13 en SelectKBest y elige el número de características que crees más adecuado.

Texto

Descripción generada automáticamente

Probando k-values en el archivo sklearn-mod3.py desde 1 hasta 13, el mejor valor de k es 3 con un score de 0.43000298317955793.

Se ha probado con 10 valores distintos de K, y se ha escogido el valor que ha dado un mayor coeficiente de determinación R-cuadrado, ya que se considera como mejor.

1. Cuál es el mejor modelo, LIN, ¿SVR o RandomForest?

¿Y si se emplean todas las variables en lugar de la más dependiente, el resultado es mejor o peor?

Según el archivo sklearn-2.py (modificado):

MSE Linear Regression (LIN): 41.828958072164035

MSE Support Vector Regression (SVR): 31.964774876168143

MSE Random Forest (RNF): 45.31977462550794

**El mejor modelo es SVR con un MSE de 31.964774876168143**

MSE Linear Regression (LIN) con todas las características: 34.705255944524936

MSE Support Vector Regression (SVR) con todas las características: 65.85931872419849

MSE Random Forest (RNF) con todas las características: 22.603066209568617

**El mejor modelo con todas las características es RNF con un MSE de 22.603066209568617**

Prueba de Wilcoxon (LIN vs. SVR):

Estadistico: 1.0

Valor p: 0.00390625

## PARTE 3

1. Carga el dataset Churn\_Modelling\_NANs.xls y haz los siguientes pasos:

Todo el código dado en este apartado es de pandas-ejercicio-8.py.

1. **Elimina las filas con valores perdidos.**

Texto

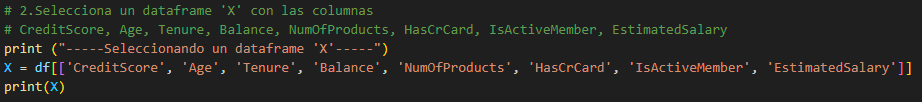
Descripción generada automáticamente

Usamos la función dropna, con axis=0, para eliminar todas las filas que contengan valores perdidos.

Axis=0 significa que eliminaremos la fila, axis=1 eliminaría la columna.

Inplace=False sirve para guardar los cambios en un nuevo objeto, pero en este caso es el mismo ya que lo igualamos a df, el cual es el objeto anterior.

1. **Selecciona un dataframe 'X' con las columnas 'CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActieMember'**



Seleccionamos un dataframe con las columnas especificadas, a partir de su header.

1. **Selecciona un dataframe 'Y' con la columna 'EstimatedSalary'**

Texto

Descripción generada automáticamente

Seleccionamos el dataframe con la columna especificada

1. **Haz tres modelos diferentes (por ejemplo, regresión lineal, SVR y Random Forest) de Y frente a X y compara el error cuadrático medio de los tres con validación cruzada (10 fold).**

**Texto

Descripción generada automáticamente**

MSE Regresión Lineal: 5.5427034190470244e-21

MSE Regresión SVR: 3171109619.6179657

MSE Random Forest: 395.73092158143334

La regresión lineal parece ajustar extremadamente bien los datos, pero es posible que esté sobreajustada o que los datos realmente sigan una tendencia lineal.

Random Forest ofrece un buen desempeño, con un MSE bajo, lo que lo hace un modelo bastante confiable.

SVR no parece ser adecuado para este conjunto de datos, ya que su error es muy elevado.

1. **(Opcional) Selecciona un dataframe 'XC' con las columnas 'CreditScore', 'Age',' Tenure', 'Balance', 'NumOfProducts',' HasCrCard', 'IsActiveMember', 'EstimatedSalary' y un dataframe 'C' con la columna 'Exited'. Haz tres clasificadores diferentes de C frente a XC y compara sus porcentajes de aciertos con validación cruzada (10 fold).**

Todo el código dado en este apartado es de pandas-ejercicio-8.1.py.

Texto

Descripción generada automáticamente

Accuracy Regression Logistica: 0.8066666666666666

Accuracy SVC: 0.7959595959595959596

Accuracy Random Forest: 0.8540404040404039

Random Forest es el modelo más preciso en este caso, logrando el porcentaje más alto de aciertos.

Tanto la regresión logística como SVC también obtuvieron buenos resultados, pero están un poco por debajo del Random Forest.

1. **(Opcional) En lugar de eliminar las filas con valores perdidos, prueba diferentes métodos de imputación y repite los puntos (4) y (6). Discute las diferencias en los resultados.**

Todo el código dado en este apartado es de pandas-ejercicio-8.2.py. No he hecho captura de pantalla debido a que no me cogía el código entero.

* Eliminación de filas con valores perdidos (dropna)

MSE Linear Regression: 3,275,714,215.13

R-squared Linear Regression: -0.0025

MSE Random Forest: 3,532,881,723.74

R-squared Random Forest: -0.0812

* Imputación por la mediana (numéricas) y moda (categóricas)

MSE Linear Regression: 3,308,343,272.50

R-squared Linear Regression: -0.0026

MSE Random Forest: 3,629,864,572.55

R-squared Random Forest: -0.1000

* Imputación por la media (numéricas) y moda (categóricas)

MSE Linear Regression: 3,308,106,177.73

R-squared Linear Regression: -0.0025

MSE Random Forest: 3,631,914,255.78

R-squared Random Forest: -0.1007

* Rellenar valores perdidos (numéricos) con ceros

MSE Linear Regression: 3,313,006,296.26

R-squared Linear Regression: -0.0025

MSE Random Forest: 3,616,467,564.12

R-squared Random Forest: -0.0944

Observaciones:

En general, los modelos muestran un rendimiento bastante bajo, con valores negativos de R-squared, lo que indica que el modelo no se ajusta bien a los datos.

La imputación por la media y la moda muestra resultados ligeramente mejores en comparación con la eliminación de filas, pero aun así los resultados son deficientes.

La imputación por ceros también produce resultados similares a los anteriores, lo que sugiere que este enfoque puede no ser efectivo en este caso específico.

1. **(Opcional) Cuál crees que es la variable que más influye en el modelo del salario? ¿Y en el modelo de la tasa de abandono?**

* Variables que pueden influir en el salario:

*Experiencia laboral*: A mayor experiencia, generalmente, mayor salario.

*Educación*: Los niveles educativos más altos suelen correlacionarse con salarios más altos.

*Puesto o rol*: Diferentes puestos tienen diferentes escalas salariales.

*Sector de la industria*: Algunas industrias tienden a pagar más que otras.

* Variables que pueden influir en la tasa de abandono:

*Satisfacción laboral*: La insatisfacción puede llevar a una mayor tasa de abandono.

*Condiciones de trabajo:* Horarios, carga y ambientes laborales pueden influir.

*Compensación y beneficios*: La percepción de la justicia en salarios y beneficios puede afectar la decisión de permanecer o abandonar.

*Oportunidades de crecimiento*: La falta de oportunidades de promoción puede llevar al abandono.

# PRACTICA 2

## PARTE 1

1. Eliminación de variables con poca varianza.

Para Variance Threshold, seria así:

Texto

Descripción generada automáticamente

Repetimos esto utilizando SelectPercentile, f\_classif y SelectPercentile y f\_classif

juntos. El proceso es análogo.

Texto

Descripción generada automáticamente

1. Eliminación de variables basada en estadísticos univariantes.

Texto

Descripción generada automáticamente

Eliminación de variables basada en estadísticos univariantes:

Columnas seleccionadas kbest chi2(iris dataset): Index (['petal\_length', 'petal\_width'], dtype='object')

Columnas seleccionadas kbest f\_classif (iris dataset): Index (['petal\_length', 'petal\_width'], dtype='object')

Columnas seleccionadas percentile chi2(iris dataset): Index(['petal\_length'], dtype='object')

Columnas seleccionadas percentile f\_classif (iris dataset): Index(['petal\_length'], dtype='object')

1. Eliminación recursiva de variables

Captura de pantalla de computadora

Descripción generada automáticamente

[1 1 1 1]

[ True True True True]

Columnas seleccionadas recursivamente Index (['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width'], dtype='object')

1. Eliminación de variables usando SelectFromModel.

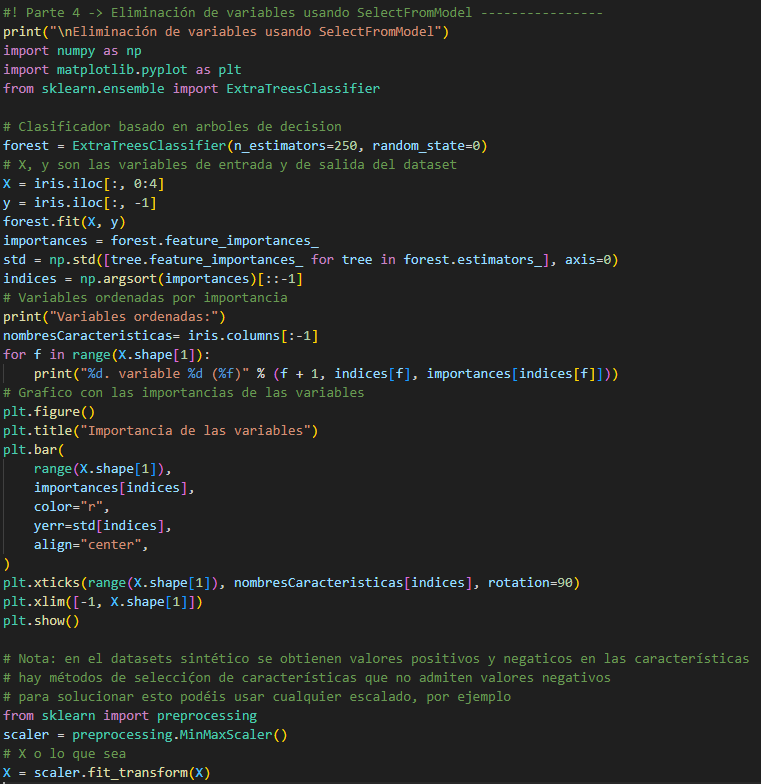
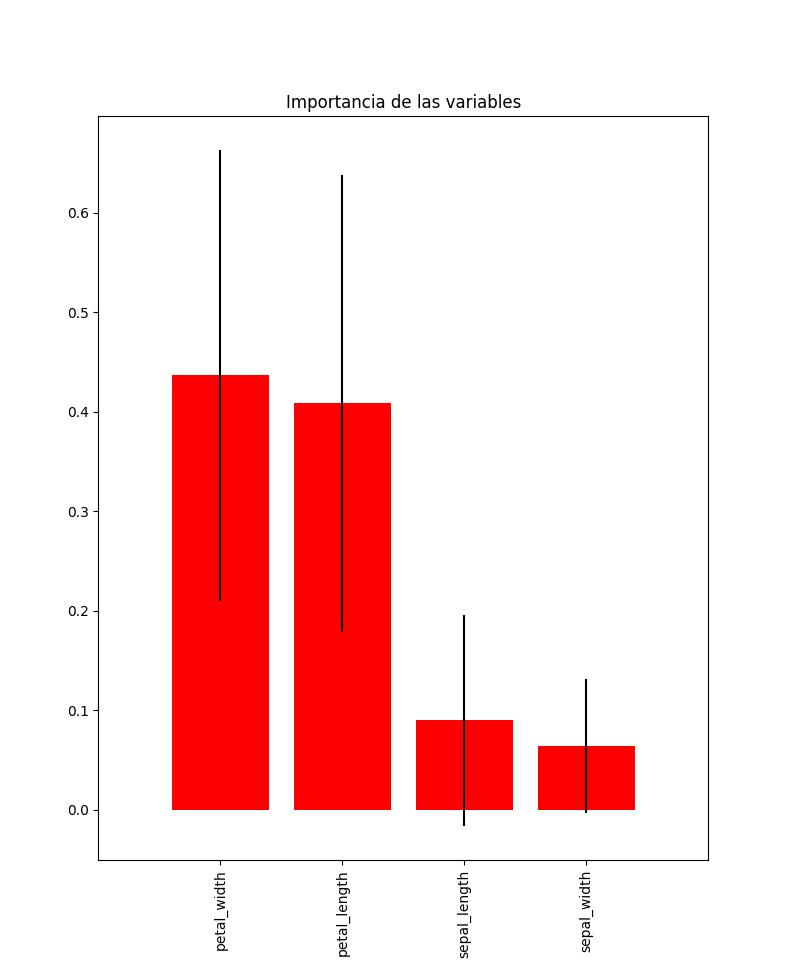


Gráfico para Iris.



Todo el proceso se ha realizado con el archivo P2\_iris.py. También se adjunta el archivo P2\_letter.py y P2\_sintetico.py con su proceso igual al iris.

Aquí están sus gráficos:

Gráfico para Letter.

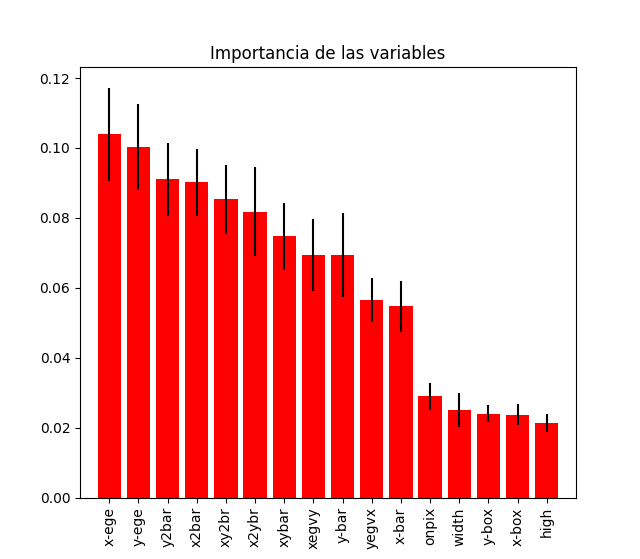
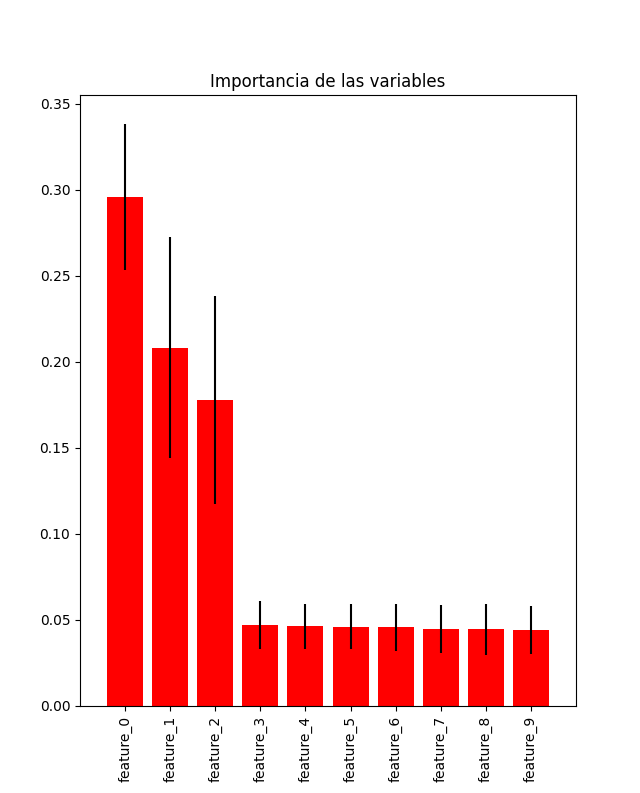


Grafico para Sintetico.



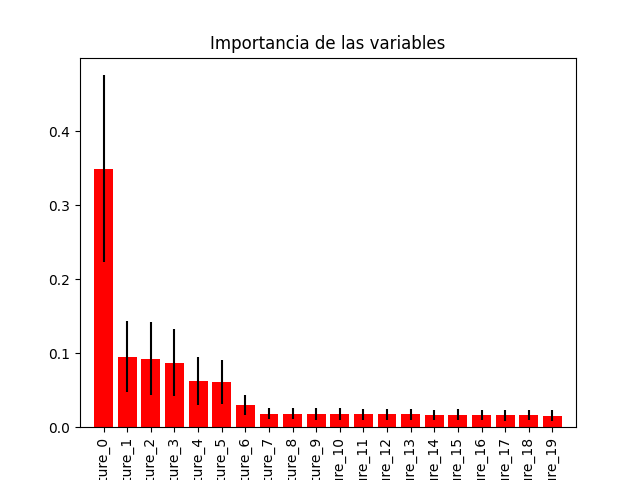
Para ver cómo se comporta el dataset sintético, intentamos añadir 20 características. De las cuales 5 son informativas, 2 son redundantes, y el resto no aportan información útil sobre la variable objetivo (y).

El objetivo del código es generar un conjunto de datos sintético y aplicar diferentes técnicas de selección de características para identificar las más importantes para un modelo de clasificación.

El código se encuentra en el archivo P2\_sintetico\_bis.py.

Texto

Descripción generada automáticamente



Los resultados muestran que un pequeño conjunto de características es responsable de la mayor parte de la predicción. Esto sugiere que es posible simplificar el modelo utilizando solo estas características clave sin perder demasiada precisión, lo que puede ayudar a reducir la complejidad y mejorar la interpretabilidad del modelo. Es decir, da igual poner más características.

# PRACTICA 3

El código para esta práctica se encuentra dentro de la carpeta Practica 3, practica3.py

1. Limpiar el dataset, quitando los valores perdidos.

Primero leemos el dataset como en las practicas anteriores:

Texto

Descripción generada automáticamente

Ahora con dropna quitamos las filas con valores perdidos.

Texto

Descripción generada automáticamente

1. Escalar o normalizar variables.

Texto

Descripción generada automáticamente

Utilizamos, por ejemplo, StandardScaler para escalar las variables.

1. Detectar las variables irrelevantes o redundantes.

Texto

Descripción generada automáticamente

Utilizamos el algoritmo KBest para seleccionar las columnas irrelevantes o redundantes.

1. Construir un modelo lineal y otro con random forest.

Texto

Descripción generada automáticamente

Construimos un modelo lineal dividendo el dataset en datos de entrenamiento y datos de prueba, y evaluamos el rendimiento del modelo mediante el error cuadrático medio y el R-cuadrado.

Se realiza de igual forma para el algoritmo de Random Forest, usando un RandomForestRegressor.

Texto

Descripción generada automáticamente

1. Realizar la validación cruzada de ambos modelos y decidir cuál es la precisión del modelo conseguido.

Texto

Descripción generada automáticamente

Precisión de modelo lineal (RMSE promedio): 0.5328384183523021

Precisión de modelo Random Forest (RMSE promedio): 0.6012690267876921

**El mejor modelo es: Modelo Lineal**

# PRACTICA 4

El código para esta práctica se encuentra dentro de la carpeta Practica 4, comercializadora.ipynb.

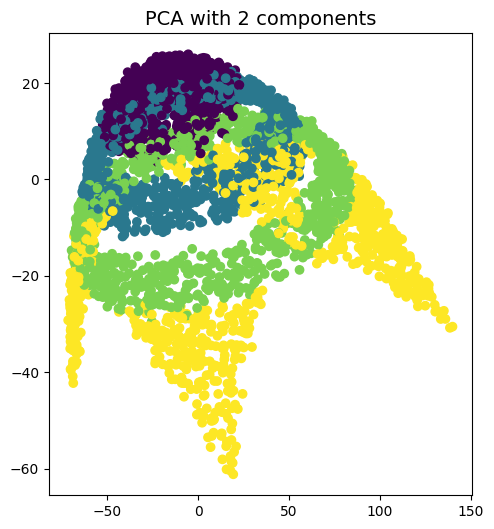
## PARTE OBLIGATORIA

Tomando como base el notebook manifold-learning.ipynb visto en clase de teoría (se incluye una copia en este enunciado, puedes cortar y pegar de este notebook lo que necesites), crea un notebook llamado comercializadora.ipynb donde se le apliquen al problema de la comercializadora los métodos indicados a continuación.

1. PCA

Texto

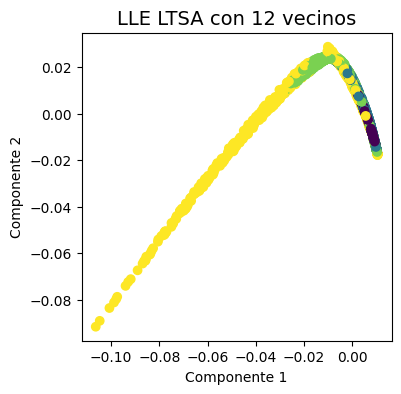
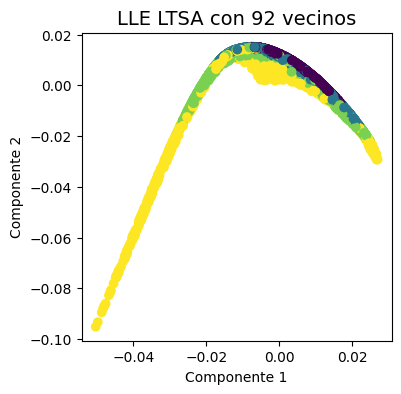
Descripción generada automáticamente



1. LLE LTSA

Texto

Descripción generada automáticamente



Gráfico, Gráfico de líneas

Descripción generada automáticamente

Como se puede observar en el gráfico, a mayor número de vecinos mayor error.

El número óptimo de vecinos es: 12 con un error de reconstrucción de 0.0000.

* Demasiadosvecinos significan que LLE intenta ajustar cada punto usando datos que no están en su vecindario local, lo cual viola la suposición de linealidad local del algoritmo.
* Al usar un número pequeño de vecinos, se respeta la estructura local y, por lo tanto, la reconstrucción es más precisa.

1. MDS

Texto

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

1. UMAP

Texto

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente

## PARTE OPCIONAL 1

1. SAMMON

Texto

Descripción generada automáticamente

Este código no es correcto, pero lo adjunto igualmente para su revisión.

1. ISOMAP

Texto

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

1. T\_SNE

Texto

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

## PARTE OPCIONAL 2

1. Kernel PCA utilizando los kernels Sigmoide y RBF

Texto

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente