Universitat Autònoma de Barcelona

PRÁCTICA 1-APC

REGRESIÓN

Grupo A403-1130

Adrià Orozco Lorente 1490952

Arnau Cruz Gargallo 1494996

INDICE

Resumen, palabras clave e introducción……P.2

Apartado B……P.3-11

Apartado A……P.12-24

EDA……P.12-14

PREPROCESSING……P.14-19

MODEL SELECTION……P.19-20

CROSS-VALIDATION……P.20-21

METRIC ANALYSIS……P.21-23

HYPERPARAMETER SEARCH……P.24

Conclusiones……P.25

Práctica 2 de Aprendizaje Computacional (Modelos de clasificación)

Adrià Orozco Lorente 1490952 y Arnau Cruz Gargallo 1494996

PALABRAS CLAVE

Clasificador
Regresión logística
SVM
KNN
Kernel
Modelo
Algoritmo
Precisión
Correlación
Outliers
Overfitting
Underfitting

RESUMEN

Durante el desarrollo de esta práctica se observa la aplicación de modelos de clasificación haciendo énfasis en la aplicación de diferentes clasificadores utilizando la regresión logística i las máquinas de vectores de soporte, aunque en nuestro caso también usaremos el knn explicado posteriormente en momento de su utilización de manera que haciendo uso de estos tres clasificadores se entiendan las mejoras que producen los kernels. También se busca evaluar correctamente los errores del modelo visualizando los datos i por supuesto el modelo resultante.
Por otra parte se busca la capacidad de aplicar técnicas de clasificación en casos reales, validando los resultados en datos reales i fomentando la capacidad para presentar resultados técnicos de aprendizaje computacional de forma adecuada delante de otras personas

INTRODUCCIÓN

Durante el desarrollo de esta práctica y bien como se ha explicado en el resumen se busca como principal objetivo la aplicación de diferentes modelos de clasificación por tal de evaluar su error i visualizar los datos y el modelo resultante. Por otra parte se procede a realizar estas técnicas de clasificación en casos reales validando además los resultados a partir de los datos de estos propios casos.

Esto se dividirá en dos apartados (B y A), donde en el primero de ellos se realizara una comparativa de modelos de una base de datos preestablecida de sklearn donde se aplicaran varios modelos de clasificación y se mostraran los resultados por tal de realizar una comparativa de modelos y de los resultados con la aplicación de funciones kernel.

Para el segundo apartado, se realizara un estudio completo de una base de datos asignada (en nuestro caso una base de datos referente a predicciones de lluvia en Australia) a partir de la cual se debe realizar un exhaustivo estudio donde se analicen todos los datos, estableciendo una variable objetivo y unas variables predictoras, para posteriormente preparar estos datos para completar un problema de clasificación.

APARTADO B

Para el apartado B se escoge la base de datos breast_cancer de sklearn para posteriormente realizar varios modelos de clasificación sobre ella y evaluar el comportamiento de estos, además del valor sobre el error que obtenemos en cada caso por tal de realizar un análisis exhaustivo de cada modelo de clasificación.

Primeramente se crea archivo jupyter notebook que cargara dicha base de datos juntamente a las librerías necesarias para aplicar los algoritmos y donde se procederá a realizar el estudio.

Así, de esta manera previamente a la carga de la base de datos comenzamos importando las siguientes librerías:

Sklearn: Esta librería sirve para realizar el aprendizaje automático, y nos ayudara a realizar el análisis predictivo. Incluye varios algoritmos de clasificación, regresión y análisis de grupos.

Numpy: Esta librería nos da soporte para crear vectores y matrices de una dimensión considerable y multidimensional, está creada precisamente para procesar grandes cantidades de datos de la forma más óptima posible y contiene una larga colección de funciones matemáticas de alto nivel para operar con ellas.

Pandas: Esta librería es una extensión de la librería de numpy, hecha para realizar la manipulación y el análisis de datos en python, ofreciéndonos así estructuras de datos y operaciones para manipular tablas numéricas y series temporales, de manera que con esta librería podremos importar de forma sencilla nuestra base de datos.

Matplotlib: Esta librería consigue la generación de gráficos a partir de datos contenidos en listas, los cuales habrán sido importados en nuestro caso previamente con uso de las funciones de la librería pandas, de manera que podamos representar los datos deseados de manera gráfica usando varios tipos de técnicas.

Scipy: Esta librería nos será útil para aplicar módulos de optimización, algebra lineal, e integración de datos entre otras, cabe destacar que es parte del conjunto de la biblioteca numpy y extiende bibliotecas de computación científica.

Seaborn: Esta librería consiste en ofrecer herramientas para la visualización de datos para Python desarrollada sobre matplotlib ofreciendo una interfaz de alto nivel para la creación de gráficas para la visualización del resultado.

Con todo esto ya se puede cargar la base de datos y realizar un análisis de datos básicos para conocer la estructura y sus características

Observando las características de la base de datos cargada, podemos observar que la variable objetivo es un valor binario (0 si es maligno, 1 si es benigno) por el cual se intentaran clasificar cada muestra, de manera que se intenta predecir si un elemento x, pertenece a la clase 0 1.

Para la realización de los modelos de clasificación se realizaran la regresión logística, svc y knn.

La regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes o predictoras.

En el caso de svc para sklearn es equivalente a una máquina de vector de soporte son un conjunto de algoritmos de aprendizaje automático supervisado que producen varias

predicciones en base a cada modelo y se juntan para obtener una única predicción. Hay que tener en cuenta que se pueden usar varios tipos de función kernel con objetivo de representar los datos en dimensiones superiores a la original de forma óptima y sin necesidad de tener que calcular los datos en un espacio superior, de forma que se realizan también comparaciones por los resultados con diversos tipos de funciones kernel.

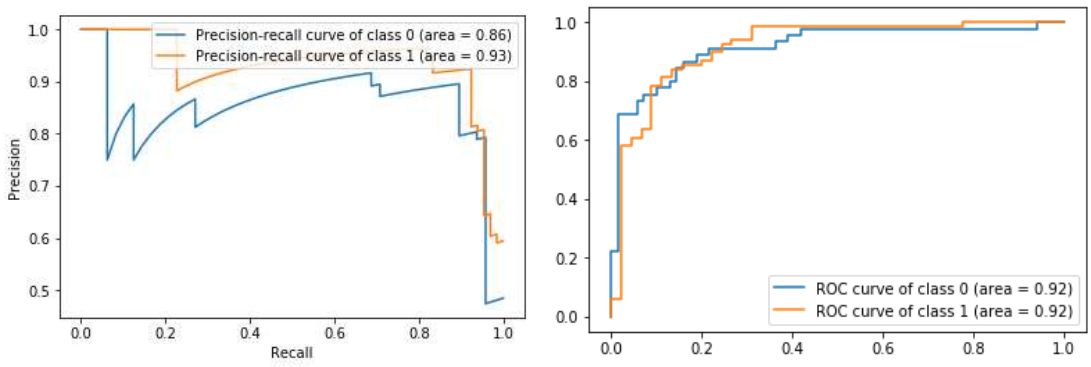
El knn es un método de clasificación supervisado que sirve para estimar la función de densidad F(x/Cj) de las predictoras x por cada clase Cj. Este es un método de clasificación no paramétrico, que estima el valor de la función de densidad de probabilidad o directamente la probabilidad a posteriori de que un elemento x pertenezca a la clase Cj a partir de la información proporcionada por el conjunto de prototipos.

Conociendo esto y los atributos que forman nuestra base de datos comenzamos aplicando una regresión logística y un algoritmo de clasificación de vectores de soporte con kernel rbf por varias particiones de los datos de entrada y observamos la correcta clasificación de estos por tal de hacer la primera comparación. Destacar antes que el kernel rbf es el kernel predeterminado utilizado dentro de las máquinas de vectores de soporte y es el más popular entre ellos.

Como primeras observaciones se debe comentar que se están utilizando pequeños porcentajes de la estructura de datos y se están teniendo en cuenta los dos primeros atributos para realizar el estudio por lo tanto puede afectar tanto al desempeño del kernel como a la evaluación del resultado causante de los datos obtenidos para el entrenamiento.

Con esto obtenemos que el mayor porcentaje de acierto lo obtiene el regresor logístico por un 0,5% de datos consiguiendo así un 89% de acierto con diferencia de 5% sobre SVM, a medida que aumentamos el porcentaje se ve la afectación en los resultados mínimamente debido a la inclusión ya sea de muestras difícilmente clasificables, outliers e incluso que estas muestras sean más fácilmente separables en un espacio dimensional superior al que ya teníamos cosa que consigue realizar SVM mediante la función kernel. Así que para el 0,7% de los datos los resultados en ambos casos empeoran cerca de un 3-5, pero añadiendo una 0,1 por ciento extra a los datos para el modelo, la máquina de vectores de soporte es capaz de mejorar su rendimiento igualando a la regresión logística.

Para continuar con la comparativa de modelos podemos considerar todas las clases en conjunto en una sola curva (micro-averaging) por tal de visualizar gráficamente los resultados del modelo SVC, además de poder representar su curva ROC la cual nos da la representación de la razón o proporción de verdaderos frente a la razón o proporción de falsos positivos según se varía un umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo).



Observando estos casos podemos ver la compensación entre la precisión y exhaustividad para diferentes umbrales, de manera que para la primera se observa hasta un 0.7 de exhaustividad aproximadamente un alto grado de precisión y exhaustividad relacionado así con la tasa baja de falsos positivos y una tasa baja de falsos negativos. Mientras que para la otra clase se produce una tasa mayor de falsos negativos desde el principio auqneu para los alores comprendidos entre 0,4 y 0,8 esta muestra una buena proporción. A partir de este valor hacia arriba se observa sobretodo un incremento de falsos negativos en ambos casos.

Para la curva ROC podemos observar una correcta proporción de verdaderos y falsos positivos indicándonos que nuestro modelo tiene una alta precisión para los valores de entrada que ha obtenido siendo esta curva proporcional al número de datos registrado, ya que a medida que aumentan los falsos negativos siguen estando los valores de verdaderos positivos muy altos para ambos modelos (Regresor logístico y svm).

Con este análisis de ambas gráficas, se procede a la realización del modelo svm con varios tipos de funciones kernel y variaciones entre las variables slack.

Para explicar la afectación del kernel en las SVM Se tiene que tener en cuenta que en muchos casos los grupos a clasificar no serán linealmente separables en el espacio original, por lo tanto una solución que ofrece la Maquina de vectores de soporte es aumentar la dimensión de los datos, la cual se puede transformar combinando o modificando cualquiera de sus dimensiones. Para hacerlo se utiliza el kernel que se una función que devuelve el resultado del producto entre dos vectores realizado en un nuevo espacio dimensional diferente al espacio original en el que se encontraban.

De forma que nos permite operar en el espacio de características original sin calcular las coordenadas de los datos en un espacio de mayor dimensión, ofreciéndonos en esencia una forma más eficiente y menos costosa de transformar los datos en dimensiones más altas.

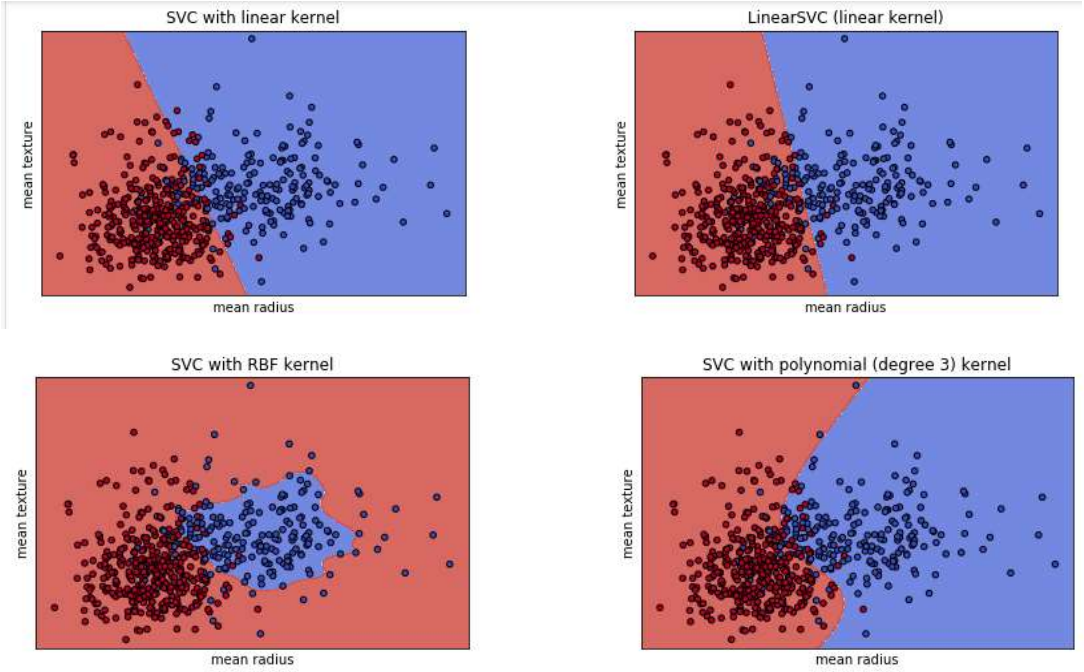
Además el uso de variables slack nos sirve en casos en los que los datos del mundo real están desordenados y casi siempre habrá algún caso que el clasificador no puede acertar, puesto que realizar una separación perfecto no siempre se posible, y en el supuesto de que lo sea, el resultado del modelo puede no ser generalizado por otros datos (overfitting). Por lo que para solucionar este problema y permitir cierta flexibilidad las svm utilizan un parámetro C que controla la compensación entre errores de entrenamiento y los márgenes rígidos creando así un soft-margin que permite algunos errores en la clasificación a la vez que los penaliza.

Cuando esta C es pequeña, los errores de clasificación tienen menos importancia y el enfoque se encuentra en maximizar el margen, mientras que cuando C es grande, el enfoque se encuentra en evitar la clasificación errónea a expensas de mantener el margen pequeño.

Trata de un compromiso, obtener un mejor clasificador y más robusto a expensas de un margen amplio.

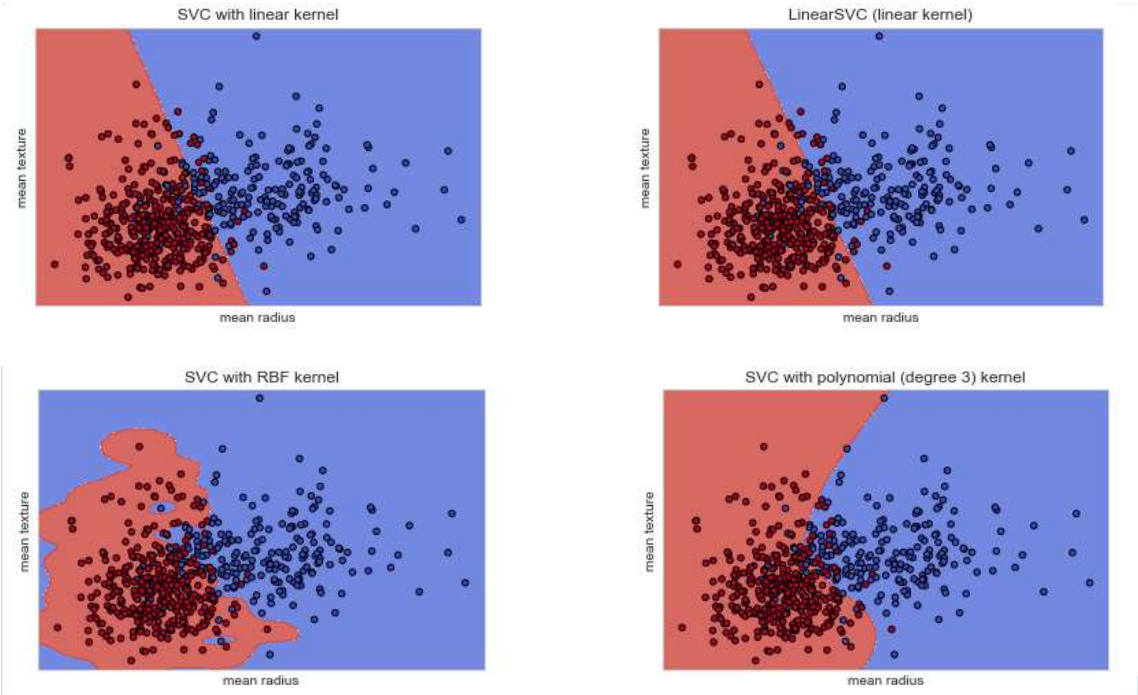
Conociendo esto se aplica el svc con kernel lineal, con rbf y polinomial con grado 3 además de un svc lineal con el mismo kernel, sobre el conjunto total de datos teniendo en cuenta las mismas variables que en los modelos anteriores.

Como primer paso establecemos 0,1 como valor para la variable slack haciendo así que los errores tengan menos importancia y el enfoque se encuentre en maximizar el margen de manera que obtenemos los siguientes resultados:



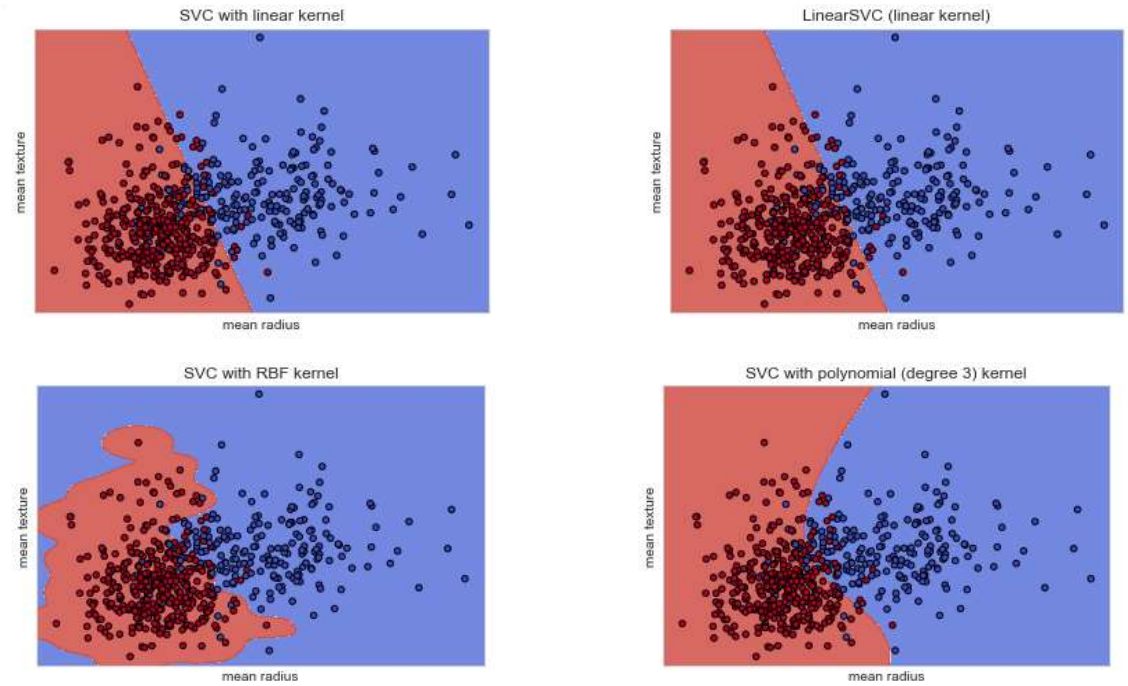
Para comenzar el primer análisis entre modelos cabe destacar que svc con kernel lineal y svc lineal con el mismo kernel producen resultados similares aunque en este caso parece que linearSVC nos ajusta un poco más el límite de decisión de manera que permite menos errores y los datos están mejor clasificados. En caso del kernel RBF el cual es el más popular y más usado, en general vemos que con un valor de la variable slack bajo obtenemos unos límites de decisión muy amplios de manera que los puntos más dispares en el grafico no son correctamente clasificados. Y para el SVC con kernel polinomial vemos que la curva de decisión se ajusta correctamente a los datos a clasificar permitiendo algunos errores ya que todavía el margen es muy amplio.

Ahora veremos qué es lo que ocurre si nos concentramos en minimizar el margen y evitar la clasificación errónea de los datos optando por un valor de C elevado, en este caso 5.



Ahora, se debe tener en cuenta que se busca evitar una clasificación incorrecta a costa de mantener márgenes todavía más estrictos, (cabe destacar que en estos casos puede ocurrir el overfitting es decir que el algoritmo utilizado no generalice correctamente haciendo casi una memorización de la solución incluso modelando el ruido de esta).

En este caso y comparando los resultados gráficamente entre los modelos similares con la varianza del coeficiente C, observamos que para el modelo SVC con kernel lineal no apreciamos diferencias significativas, mientras que para el LinearSVC que previamente parecía ligeramente una mejor opción, ahora clasifica mejor la parte de datos que están más condensados y es más difícil realizar una correcta clasificación pero eso ocurre a expensas de producir algún error más en valores que antes predecía bien, la solución ante esto sería equivalente a usar una función polinomial ya que una recta no es capaz de modelar la curva natural de los datos y cómo podemos observar al aumentar este coeficiente y usar el kernel polinomial, aquellos valores que se encontraban en los límites de decisión están ahora correctamente clasificados y además se producen mayores aciertos en los valores que son más difíciles de clasificar aumentando ligeramente la precisión de este modelo. Por ultimo con el kernel RBF el cual como ya se ha comentado es el más popular entre todos, obtenemos una gran mejora del algoritmo consiguiendo muy pocos errores y siendo capaz de modelar incluso parte de los outliers de los datos, cosa que puede indicar overfitting. Para corregir esto vamos a probar con un valor más bajo de la variable slack de manera que no clasifique los outliers y que sea capaz de modelar correctamente los demás datos, y gracias a esto obtenemos que con un valor de C equivalente a 0,9. Donde vemos que seguimos manteniendo en cada uno de los modelos los menores errores de clasificación.



Una vez realizado y comparadas las diferencias entre las predicciones de las máquinas de vectores de soporte con diferentes kernels, se procede a la realización de los tres algoritmos comentados al inicio de esta memoria con un conjunto de datos reducido y se comparan los resultados entre sí.

Para ello realizamos una regresión logística con los dos conjuntos ya entrenados previamente tanto de train y de test que contienen un 0,8 % del tamaño del dataset el cual es equivalente a unas 114 muestras y así comparar los resultados de cada modelo mediante la función accuraccy_score y classification_report (Destacar que se usara el kernel predeterminado para el modelo svc y se usara 8 como número de vecinos para el knn como valor predeterminado)

Regresión Logística:

```
Train Set Accuracy:88.79120879120879
Test Set Accuracy:90.35087719298247

Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.93      0.84        30
           1       0.97      0.89      0.93        84

   micro avg       0.90      0.90      0.90       114
   macro avg       0.87      0.91      0.88       114
weighted avg       0.92      0.90      0.91       114
```

Para este tipo de regresión que ya de antemano sabemos que es un clasificador muy válido para problemas de clasificación binaria como es este caso, obtenemos una precisión de 88% en el conjunto de entrenamiento y un 90% en el conjunto de test, claro indicador de que nuestro modelo está generalizando bien debido a que no hay una mala precisión al introducir nuevos atoa al algoritmo entrenado consiguiendo incluso una mejor precisión con nuevos datos de entrada (No hay overfitting), y tampoco underfitting porque con una precisión de este calibre podemos ver que la función logística se ajusta a la curva natural de los datos en este caso.

Por otra parte analizando los resultados a fondo podemos observar que los valores 1 (benigno) tienen cerca de un 100% de precisión en cuanto a la clasificación, mientras que los valores 0 (maligno), bajan su precisión hasta el 76%, si bien no es una mala aproximación, nos indica que habrá valores límite difíciles de clasificar, esto también es debido a que el número de muestras que se obtienen para el análisis de estos datos en este caso es menor en proporción por lo que el entrenamiento está más especializado en los valores 1.

Observando el resultado en los valores de recall, que nos indica el ratio de verdaderos positivos con falsos negativos, en ambos valores están en un 90% de precisión por lo que obtenemos una buena puntuación para este apartado y este modelo con los datos que se han introducido para el entrenamiento.

Por ultimo observando los parámetros de f1-score el cual nos indica una medida de la precisión del modelo sobre el dataset entrenado obtenemos un 93% de precisión para los valores malignos y un 84% ara los valores benignos, lo cual nos indica que el modelo entrenado realiza buenos resultados, y la disminución del porcentaje de precisión respecto a los valores benignos se debe a que hay un número menor de muestras a clasificar que de valores malignos, lo cual afecta a que sea más difícil generalizar el modelo para esos valores, aunque igualmente no produce una mala relación precisión-recall.

SVC:

```
Train Set Accuracy:94.94505494505493
Test Set Accuracy:87.71929824561403

Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.85      0.80        33
           1       0.94      0.89      0.91        81

   micro avg       0.88      0.88      0.88       114
   macro avg       0.85      0.87      0.86       114
weighted avg       0.88      0.88      0.88       114
```

En este caso, recordemos que ya se había analizado previamente el impacto de las funciones kernel y las variables de holgura sobre este tipo de modelo de aprendizaje automático supervisado, y en este caso estamos realizando el kernel predeterminado para comparar entre los demás modelos.

Primeramente podemos observar que se consigue una precisión muy elevada para el conjunto de entrenamiento estando muy cerca del 95%, indicativo de que el entrenamiento del modelo ha funcionado de forma óptima y es capaz de clasificar un gran número de muestras de forma correcta, pero sin embargo el resultado para el conjunto de test es algo peor, siendo este un 7% inferior. Teniendo en cuenta que el conjunto total de muestras es un numero de rango inferior este porcentaje puede tenerse en cuenta como elevado, lo cual es un indicativo de que el modelo pueda sufrir overfitting, para corregir esto deberíamos hacer uso de variables slack e incluso probar otro tipos de funciones kernel en función de la distribución de la nube de puntos generada, con objetivo de que se adapte mejor a los datos nuevos, sin memorizar la solución ni el ruido del conjunto de entrenamiento.

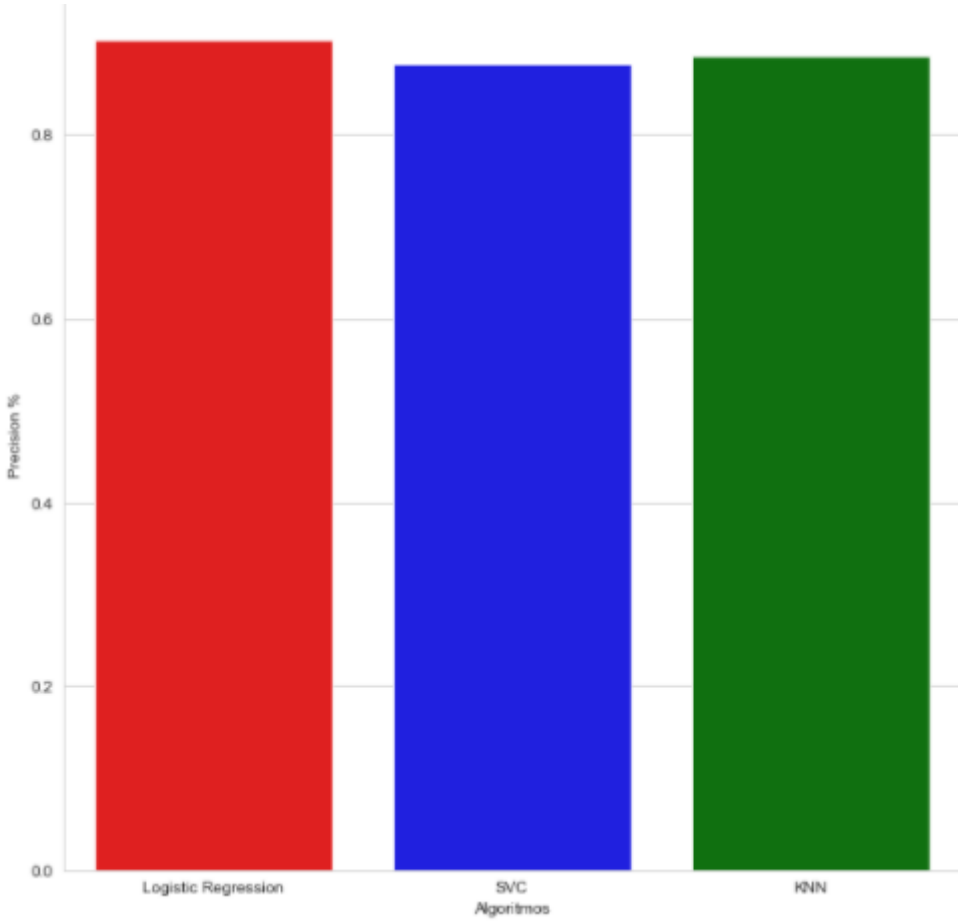
Observando los porcentajes obtenidos para los valores benignos y malignos, ocurren cosas similares a la regresión logística, donde al tener un número menor de muestras malignas, estas obtienen un peor porcentaje de precisión que en las benignas, mientras que la diferencia reside respecto a los ratio de verdaderos positivos contra falsos negativos, que en este caso disminuyen un pequeño porcentaje respecto a la regresión logística (debido al overfitting previamente comentado), al igual que la medida de la precisión del modelo la cual es más baja en ambos casos sobre todo por las muestras malignas la cual en este caso es del 80%.

KNN:

```
Train Set Accuracy:90.76923076923077
Test Set Accuracy:88.59649122807018

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.85      0.82        34
           1       0.94      0.90      0.92        80

   micro avg       0.89      0.89      0.89       114
   macro avg       0.86      0.88      0.87       114
weighted avg       0.89      0.89      0.89       114
```

En el caso del KNN observamos un patrón similar al que estaba ocurriendo con el SVC, pero con resultados mas aproximados al regresor logístico, siendo en este caso un 2% menor la precisión del conjunto de test sobre el conjunto de entrenamiento, dándonos la idea de que también es posible que el modelo generalize peor. Esto se debe al número de clases establecido para la clasificación, la cual afecta al algoritmo y hemos establecido 8 como valor predeterminado, alterando este número de clases según la distribución de puntos es posible que los resultados varíen y mejoren, pero es difícil acertar un numero de clases ideal porque cada problema tiene una naturaleza distinta, así que para saber cuál es el número de clases ideal deberíamos utilizar fórmulas matemáticas para cada problema para quedarnos con el mejor resultado y el que generalize mejor. A partir de esto si bien, no son malos resultados, se analizan los valores obtenidos por cada clase separada, donde vemos que se aumenta la precisión de la clasificación de los modelos malignos ligeramente respecto a los dos clasificadores anteriores, al igual que la clasificación de los valores benignos no mejor a la regresión lineal e iguala al modelo svc. Por otra parte los ratio de verdaderos positivos contra falsos negativos son prácticamente iguales que en el modelo svc ya que tienen resultados muy similares y empeoran ligeramente los resultados obtenidos en la regresión logística.

Por ultimo cabe destacar que la puntuación de la precisión obtenida por el modelo y este conjunto de datos en concreto en este caso para modelos malignos mejora ligeramente al modelo SVC en ambas clases, mientras se obtienen valores ligeramente por debajo respecto al represor logístico.



Como podemos observar gráficamente en la comparativa de la precisión real de los tres modelos con los parámetros establecidos que se han comentado anteriormente, podemos destacar que las observaciones realizadas están en lo correcto, donde el regresor logístico es ligeramente superior al KNN y este es ligeramente superior al SVC, por todo lo que se ha ido comentando en cada modelo.

Notar, que estos resultados varían según la naturaleza del problema, los parámetros establecidos, la regulación, las variables de holgura y el tipo de función kernel para los SVC, como el número de clases para el KNN. En este caso al ser un problema de clasificación binaria y sin haber modificado los parámetros de los modelos svc y knn, es lógico que la regresión logística pueda ser ligeramente superior para este caso en concreto, pero para otro tipo de clasificadores u otro tipo de problema de clasificación los resultados pueden variar por cada modelo, de igual forma que hubieran variado si modificamos ciertas variables y ciertos datos del conjunto de entrada para el entrenamiento. Por lo que podemos decir que no es que haya un modelo superior a otro, simplemente se debe escoger según la naturaleza del problema y la representación de los datos.

APARTADO A

En este apartado se muestra como se realiza un modelo de clasificación a partir de un conjunto real de datos equivalente a datos sobre la lluvia en Australia los cuales se analizan de forma exhaustiva y se tratan por tal de realizar un algoritmo lo más preciso y completo posible

EDA:

Para comenzar con nuestro modelo se debe examinar y entender como está estructurada nuestra base de datos, para ello podemos utilizar funciones de la librería pandas con objetivo de ver los atributos que esta contiene, las relaciones entre ellos, el tipo de variables que existen y así poder comenzar a realizar las primeras hipótesis sobre las variables que debemos incluir en el estudio.

Primeramente se carga la base de datos descargada y localizada en la carpeta Data/ApartadoA mediante un csv delimitado por comas, en una variable la cual contendrá toda esta información. A partir de aquí podemos examinar cuantos datos contiene, y como se almacenan estos.

Mediante la función shape() podemos ver el número de filas, correspondiente al número de elementos de nuestro dataset y el número de columnas**, correspondiente al número de atributos que tiene nuestra base de datos que en nuestro caso es igual a 23 atributos.**

Para saber que atributos tenemos exactamente y como estos están almacenados, podemos recurrir a varios tipos de funciones de la librería pandas, en nuestro caso hacemos uso de la función head() para observar las primeros 5 filas del dataset y así poder realizar un primer vistazo y tener una pequeña referencia sobre cada elemento, gracias al que ya primeramente podemos observar que existe algún valor nulo en cada fila los cuales tendremos que procesar más tarde en siguientes fases, además de ver datos tanto numéricos como categóricos. Para saber bien que tipos de atributos tenemos y como están representados podemos usar la función info() que nos muestra el tipo de dato almacenado en cada uno de nuestro atributos los cuales representamos seguidamente, respondiendo así a la pregunta: **¿Qué tipos de atributos tenemos?**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
Date             145460 non-null object
Location         145460 non-null object
MinTemp          143975 non-null float64
MaxTemp          144199 non-null float64
Rainfall         142199 non-null float64
Evaporation      82670 non-null float64
Sunshine         75625 non-null float64
WindGustDir      135134 non-null object
WindGustSpeed    135197 non-null float64
WindDir9am       134894 non-null object
WindDir3pm       141232 non-null object
WindSpeed9am     143693 non-null float64
WindSpeed3pm     142398 non-null float64
Humidity9am      142806 non-null float64
Humidity3pm      140953 non-null float64
Pressure9am      130395 non-null float64
Pressure3pm      130432 non-null float64
Cloud9am         89572 non-null float64
Cloud3pm         86102 non-null float64
Temp9am          143693 non-null float64
Temp3pm          141851 non-null float64
RainToday        142199 non-null object
RainTomorrow     142193 non-null object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

Estos son los principales datos que tenemos en nuestra base de datos, los cuales están divididos en object y float64, siendo los primeros **tipos de datos categóricos y los segundos tipos de datos numéricos**. Algunos de los datos categóricos son ordinales de más de 2 categorías lo cual indica que los tendremos que convertir en valores numéricos para el posterior estudio y así observar si hay posibles correlaciones entre ellos, en cambio otros son tan solo de dos categorías, los cuales pueden ser transformados en valores binarios (0, 1) y puede ser un indicativo también de que alguno de estos atributos sea una variable objetivo del problema de clasificación.

Estos atributos que acabamos de describir son RainToday y RainTomorrow. Entre estos dos se decide que el atributo objetivo será RainTomorrow, ya que nos parece más interesante estudiar la probabilidad de que llueva al día siguiente teniendo en cuenta las condiciones climáticas del día anterior según diversos factores, que no la probabilidad de que llueva hoy la cual podemos saber de forma más sencilla.

Con esto **se responde a la pregunta de cómo es el target y que categorías diferentes existen** Sabiendo que RainTomorrow es una variable ordinal de 2 categorías (Yes, No), aunque posteriormente se cambiaran esos valores por binarios para el estudio de correlaciones.

Para **poder ver si existe alguna correlación entre X e y**, convertimos las variables RainToday y RainTomorrow en variables cardinales de manera que podamos ver mediante una matriz de correlación si existe algún tipo de relación. Destacar que es difícil cuantificar este tipo de datos y extraer conclusiones a partir de una matriz de correlación. Por lo que en este caso no observamos ningún tipo de correlación en valores numéricos. Por lo que posteriormente se procede a realizar gráficos entre variables para observar de forma visual como se distribuyen los datos tratando outliers y valores nulos para ver si nos puede dar algún indicador de que existan ciertas correlaciones y volver a hacer la matriz de correlación.

Para observar las correlaciones y relaciones entre variables primeramente comprobamos cuantos valores nulos existen por categoría, haciendo uso de la función isnull().sum() de manera que obtenemos el número de nulos que existen por cada atributo.

```
Date               0
Location           0
MinTemp          468
MaxTemp          307
Rainfall           0
Evaporation    59694
Sunshine       66805
WindGustDir     9163
WindGustSpeed   9105
WindDir9am      9660
WindDir3pm      3670
WindSpeed9am    1055
WindSpeed3pm    2531
Humidity9am     1517
Humidity3pm     3501
Pressure9am    13743
Pressure3pm    13769
Cloud9am       52625
Cloud3pm       56094
Temp9am          656
Temp3pm         2624
RainToday          0
RainTomorrow       0
dtype: int64
```

Como podemos observar, existe una gran cantidad de valores nulos, cosa que afectara al estudio de las variables, correlaciones y relaciones entre variables. Así que para realizar un análisis exhaustivo de los datos trataremos estos valores nulos en el siguiente apartado de preprocessing, el cual comenzaremos con este tratamiento y el análisis exhaustivo que dejamos pendiente hasta ese instante.

Para responder a la pregunta de si **están balanceadas las etiquetas** podemos observar que estas toman valores distintos por cada atributo de manera que existen fechas, valores decimales, variables almacenadas en strings, y valores enteros, por lo que es difícil categorizar todo de forma uniforme y esto **afectará a la clasificación y distribución** debido a que se deben tomar decisiones sobre cómo se representaran estos datos y como se transformarán para realizar el estudio y poder comparar-los.

PREPROCESSING:

Para comenzar el apartado de pre procesamiento de los datos se procede a realizar **la limpieza de los valores nulos**, donde nos habíamos quedado en el anterior apartado. Debido a la gran cantidad de estos valores, se decide **en vez de borrar cada elemento, realizar una media aritmética de las columnas en cuestión y rellenar esos valores nulos con el resultado de la media calculada**, **(ya que en el caso de borrar todos nos quedaríamos con un número limitado de filas de la matriz)** para así aproximar un valor que podría ser real y obtener un conjunto mayor de datos en el momento de realizar el estudio, con lo que ahora sí que podremos realizar con mayor exhaustividad un estudio entre correlaciones y variancias entre variables. Por supuesto para aquellos valores no numéricos no se podrá realizar este método por lo que para este caso, como las variables relacionadas con la dirección del viento tienen también valores nulos, pero no podemos calcular una media entre ellos ya que son datos no cuantificables, remplazaremos todos aquellos valores por los que aparecen de manera más frecuente con la función mode()[0].

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
Day              0
Month            0
Year             0
dtype: int64
```

Como podemos ver ya no existen valores nulos

Una vez conseguido este paso se procede a mostrar la matriz de correlaciones mediante el mapa de calor para verlo de forma más gráfica y visual.

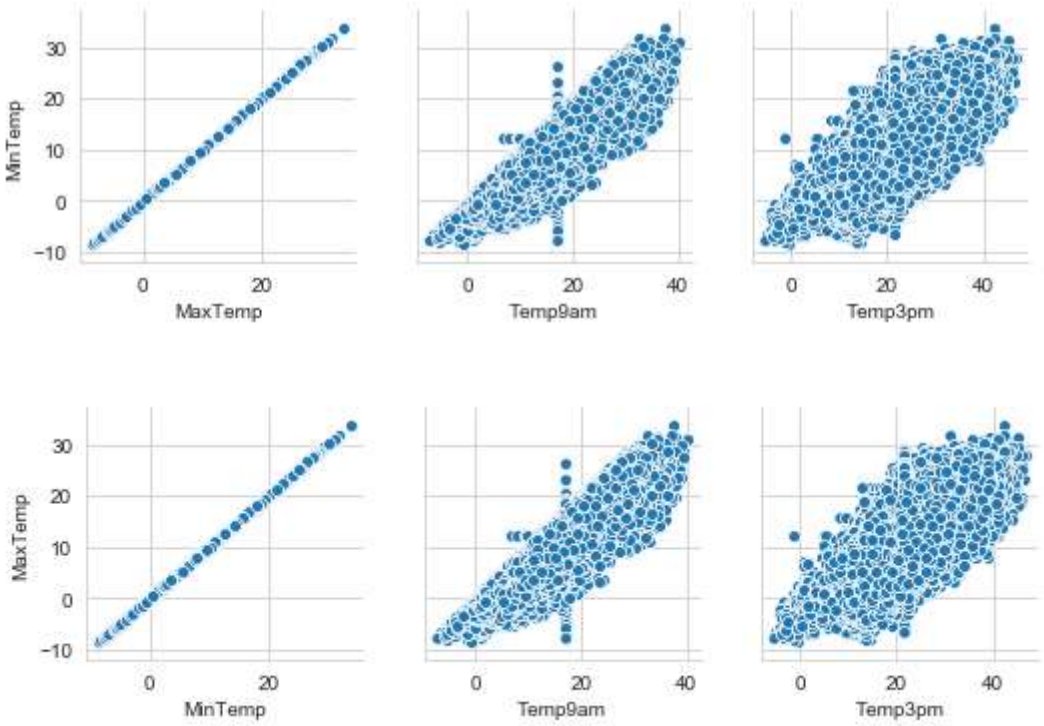
Como podemos ver no hay una correlación alta entre nuestras variables independientes y la variable objetivo escogida, esto es debido a que como se ha comentado anteriormente, se han asignado valores binarios para mostrar gráficamente nuestra variable objetivo, por lo que la relación que haya entre las demás variables cardinales es difícil de cuantificar, por lo que para ver si existen correlaciones entre X e y se deben realizar distintos tipos de gráficos para verlo de forma visual.

Si bien antes de observar la correlación con la variable objetivo podemos visualizar que existe cierta correlación entre algunas variables independientes cosa que influirá en el estudio posterior con la eliminación de variables redundantes y confusoras. Entre estas variables se encuentran la temperatura mínima con la temperatura máxima y la temperatura de las 9 de la mañana con la temperatura de las 3 de la tarde, entre las cuales existe una muy alta correlación en los 4 casos, y por otra parte la presión atmosférica de las 3 de la tarde con la presión atmosférica de las 9 de la mañana, ambas con una relación prácticamente cercana a 1.

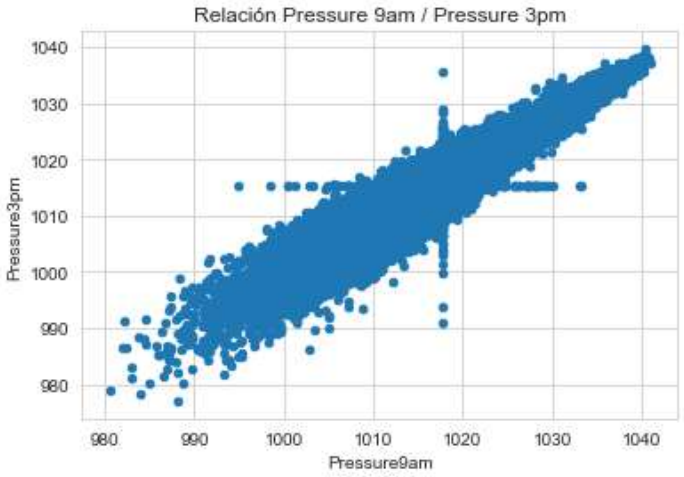
Además tiene también una elevada correlación las variables relacionadas con la velocidad del viento, por lo que también serán tenidas en cuenta.

En este instante, se realizan los gráficos para estas variables para compararlas entre si y ver las relaciones de forma gráfica con objetivo de comenzar el filtrado de datos.

Una vez realizadas todas las posibles relaciones entre variables independientes, se amplían los gráficos individualmente para mostrar las primeras conclusiones acerca de estos.

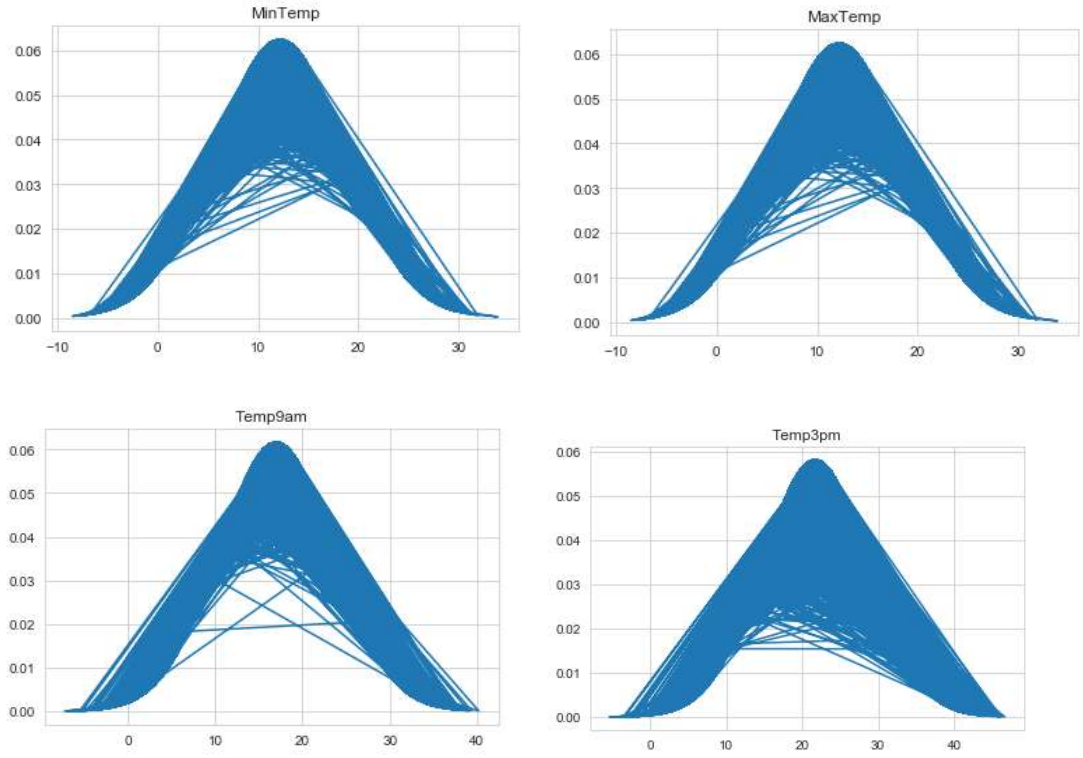


Como podemos observar la relación entre la temperatura mínima y máxima es totalmente proporcional, claro indicativo que para el estudio final podremos prescindir de una de las dos variables. Por otra parte la relación entre la temperatura mínima, máxima y la temperatura a diversas horas del día vemos que puede tener cierta proporción pero esto de momento no nos da demasiada información para el caso que queremos predecir qué es si lloverá al día siguiente.



En cuanto la presión atmosférica a distintas horas del día podemos observar que existe cierta relación proporcional entre ellas también por lo que con una de las dos variables creemos que será suficiente para realizar la posterior clasificación.

Para seguir con el filtrado de variables se realiza una grafico normalizado para ver cuales siguen una distribución gaussiana y obtenemos los siguientes resultados:



Sabiendo así que las variables que representan una distribución normal son aquellas relacionadas con la temperatura, si bien algunas otras variables como las relacionadas con la velocidad del viento presentan estructuras similares pero no se pueden considerar gaussianas.

En este punto y sabiendo que posibles variables pueden ser de utilidad para el estudio posterior debida su correlación y su distribución, pasamos al estudio de las variables que no nos son tan fáciles de cuantificar como aquellas relacionadas con la dirección del viento o la fecha en las que se toman los datos sobre el clima.

En cuanto a los datos categóricos, encontramos los atributos relacionados con la dirección del viento a las 3pm y a las 9 am (WindDir9am y WindDir3pm), Location (Relacionado con la localidad en la que se produce la extracción de los datos climáticos) Los cuales son difíciles de cuantificad debido a que son ordinales de más de dos categorías por lo cual es posible utilizar un **LabelEncoder con la función preprocessing de la librería sklearn** con objetivo de realizar la codificación de estas variables por tal de incluirlas en el estudio y poder ver si tienen relación tanto entre ellas como posteriormente con la variable objetivo, De manera que ahora queda nuestro dataset de la siguiente manera:

Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am
2	13.4	13.4	0.6	5.472516	7.63054	13	44.0	13	...	71.0	22.0	1007.7
2	7.4	7.4	0.0	5.472516	7.63054	14	44.0	6	...	44.0	25.0	1010.6
2	12.9	12.9	0.0	5.472516	7.63054	15	46.0	13	...	38.0	30.0	1007.6
2	9.2	9.2	0.0	5.472516	7.63054	4	24.0	9	...	45.0	16.0	1017.6
2	17.5	17.5	1.0	5.472516	7.63054	13	41.0	1	...	82.0	33.0	1010.8

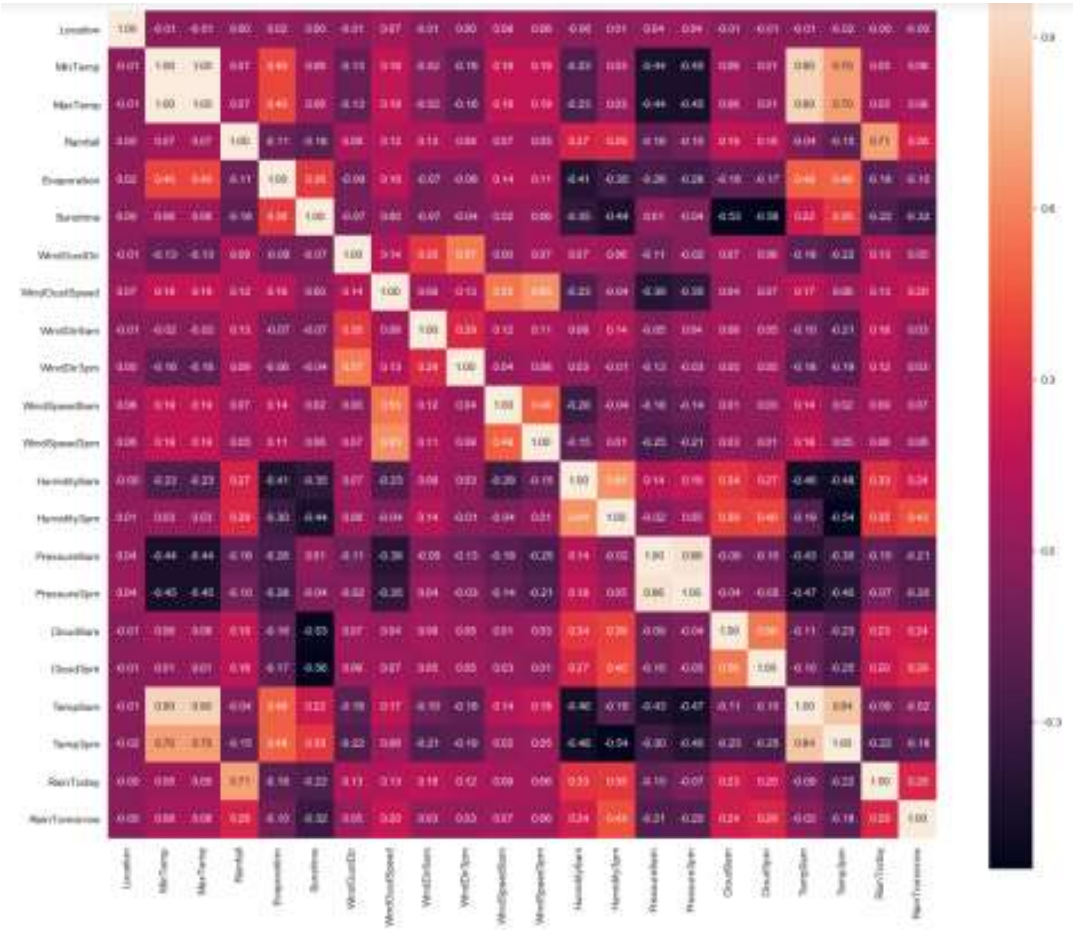
En este punto con todos os datos cuantificados, procedemos a eliminar los outliers que puedan existir en nuestros atributo, esto lo realizaremos mediante la librería scipy con la cual podremos usar la función zscore que nos devolverá el número de desviaciones típicas que un valor toma respecto a la media o muestra de los otros valores de cada atributo respecto a cada elemento, de manera que eliminamos todos esos elementos y ahora eliminamos cerca de casi un 6% de las muestras.

Como podemos observar también, **los datos no están normalizados,** ya que se comprenden valores distintos de 0-1 entre diversas variables, por lo cual sería interesante **realizar una normalización** antes de continuar el estudio por tal de aumentar la precisión en el modelo posterior pudiendo hacer uso de variables que a simple vista por ahora parecen no tener impacto en el estudio. De manera que utilizaremos una **Escala de mínimos y máximos para normalizar los datos** mediante la función MinMaxScaler de la librería preprocessing donde después una vez obtenidas estas escalas transformaremos los datos con la función transform teniendo en cuenta el resultado de la nueva escala

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity9am	Humidity3pm	Pr
0	0.041667	0.535620	0.535620	0.021739	0.364834	0.526244	0.866667	0.527778	0.866667	0.933333	...	0.670455	0.212121	
1	0.041667	0.377309	0.377309	0.000000	0.364834	0.526244	0.933333	0.527778	0.400000	1.000000	...	0.363636	0.242424	
2	0.041667	0.522427	0.522427	0.000000	0.364834	0.526244	1.000000	0.555556	0.866667	1.000000	...	0.295455	0.292929	
3	0.041667	0.424802	0.424802	0.000000	0.364834	0.526244	0.266667	0.250000	0.600000	0.000000	...	0.375000	0.151515	
4	0.041667	0.643799	0.643799	0.036232	0.364834	0.526244	0.866667	0.486111	0.066667	0.466667	...	0.795455	0.323232	

Como ahora nuestras variables están clasificadas sin outliers y todas categorizadas entre valores normalizados **no tiene sentido aplicar un PCA** ya que estas variables pertenecen todas a un mismo plano de coordenadas, en caso contrario sí que sería una buena opción **aplicar un PCA para representar una tabla de datos multivariables en un set más pequeño y concentrado.**

Con los nuevos datos realizamos de nuevo la matriz de correlaciones para observar los posibles cambios que se hayan podido producir en nuestro modelo, y así tomar ya la decisión de que variables utilizaremos para el estudio.



Procedemos a eliminar las variables que tienen alta correlación entre ellas con los nuevos valores normalizados y sin outliers tomando la decisión en función de la correlación que tengan también con el atributo objetivo, por lo tanto decidimos eliminar MinTemp ya que tiene una correlación igual con MaxTemp (1.00), Humidity9am ya que tiene una muy alta correlación con humidity3pm y está ultima tiene más correlación con la variable objetivo. Por otra parte, quitamos Pressure9am debido a los mismos motivos que en el anterior caso teniendo en cuenta la presión de las 3pm, y por ultimo eliminamos Temp3pm por los mismos motivos y su comparación con la variable temp9am.

MODEL SELECTION:

Para este problema, se ha de tener en cuenta que es un problema de **clasificación binaria por lo tanto se considera utilizar una regresión logística y un modelo relacionado con las máquinas de vectores de soporte**. Esto se decide debido a que la regresión logística es el método de referencia para problemas de regresión binaria y utiliza una función logística que puede coger cualquier valor real entre 0 y 1 pero nunca en estos límites, por lo tanto es ideal para nuestros datos normalizados. Por otra parte con una SVM cabe recordar que se desarrolló originalmente para problemas de clasificación binaria y ahora mismo es uno de los referentes en el ámbito del aprendizaje estadístico y computacional.

En caso del clasificador de vectores de soporte **consideramos utilizar los kernels lineal, RBF y polinomial,** de forma que podamos comprobar cual se adapta a la curva natural de los datos de forma óptima y precisa sin producir overfitting. En cuanto a la aplicación de los kernels **creemos que el más preciso será el RBF,** debido a que es el más popular entre ellos y el más utilizado en

este tipo de problemas de clasificación, a parte, considerando los resultados obtenidos en el apartado b por otro problema de clasificación binaria, pudimos ver como el kernel se adaptaba mucho mejor a la naturaleza de los datos ya que no seguía una estructura predefinida ni lineal ni polinómica y era capaz de adaptarse a una nube de puntos dispersa en un espacio de dos dimensiones, de manera que es capaz de maximizar el número de verdaderos positivos que se encuentren un poco alejados de los valores medianos, los cuales son más difíciles de predecir mediante kernels lineales o polinómicos, aunque si bien este último puede tener cierta mejora respecto al lineal en estos casos, el rbf puede reducir el rango de acción de los límites de decisión, consiguiendo así una mayor precisión.

En cuanto a términos de rapidez, sabemos **que el más rápido entre ambos algoritmos será el regresor logístico**, ya que en este caso el svm depende de muchos factores y tenemos demasiados atributos para considerar durante la regresión por lo tanto la complejidad de este ser muy elevada, cosa que imposibilita la realización del algoritmo de forma rápida y eficiente.

Puede ser conveniente realizar un ensemble como pueden ser bagging o boosting, tanto uno como otro son conjuntos de modelos de aprendizaje computacional, cada modelo produce una predicción diferente, y las predicciones se combinan para obtener una única predicción. Esto desempeña en una predicción general a través de varias predicciones combinadas lo que en ocasiones puede dar un mejor resultado, pero como principal inconveniente encontramos que para modelos con multitud de datos y variables independientes realizar varios modelos puede ser ineficiente debido a la alta complejidad de estos que en parámetros de tiempo puede ser muy elevada pero también en algoritmos como el bagging se puede realizar de forma paralela cosa que optimizaría el resultado

CROSS VALIDATION:

La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. La validación cruzada es una manera de predecir el ajuste de un modelo a un hipotético conjunto de datos de prueba cuando no disponemos del conjunto explícito de datos de prueba. **Por lo tanto es importante para conseguir los mejores parámetros para entrenar el modelo sin introducir casos que puedan causar overfitting o underfitting.**

Para separa los datos en conjuntos de train-test podríamos hacer uso de la función train_test_split a la cual le asignamos un valor total que viene a ser la medida de cada conjunto de entrenamiento, y un valor para que coja muestras de forma aleatoria. Cabe destacar **que mientras más muestras hayan para el conjunto de entrenamiento** mejores resultados tendrá el modelo ya que tendrá más elementos que clasificar y supervisar de manera que recopilara más información para nuevos posibles casos a clasificar posteriormente, si bien el aumento de este número de datos produce efectos notorios sobre el resultado, también afecta en el tiempo de ejecución debido al alto procesamiento de todo los elementos.

En nuestro caso separaremos los datos teniendo en cuenta un tipo de validación **cruzada k-fold** basada en k iteraciones, esta o también llamada **k-fold-cross-validation**, realiza una división de datos en k subconjuntos de manera que uno de los subconjuntos se utiliza como datos de prueba y el resto como datos de entrenamiento, el proceso se repite durante k iteraciones con cada uno de los posibles subconjuntos de prueba, para finalizar se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso ya que evaluamos k combinaciones de datos de entrenamiento y de prueba, con la desventaja que

puede ser lento computacionalmente. Lo más común es elegir una **validación cruzada de 10 iteraciones que es la que usaremos nosotros**. El hecho de escoger más o menos iteraciones **afecta en el resultado final** ya que se realizan menos casos de conjuntos de prueba y entrenamiento por lo que los resultados pueden ser menos precisos debido a que el número de pruebas se reduce al igual que el número de muestras utilizadas.

En el caso de querer utilizar **Leave-one-out** este consiste en dejar un sub-conjunto fuera de cada iteración, este enfoque puede considerarse optimo desde el punto de vista de exactitud pero a costa de un aumento de la capacidad de computación necesaria para el entrenamiento de todos los modelos que generalmente no es asumible y más en nuestro caso con tanta cantidad de variables independientes y numero de elementos

METRIC ANALYSIS:

Para realizar el análisis métrico, tenemos un conjunto de resultados que nos indican diferentes valores respecto al acierto de los modelos, entre ellos se encuentran:

Acurracy_score: Esta función calcula la precisión del subconjunto entrenado de manera que el conjunto de etiquetas predichas para una muestra debe coincidir exactamente con el conjunto de etiquetas correspondiente

F1_score: Esta se puede interpretar como una medida entre la precisión y la exhaustividad de manera que es una medida de la precisión de un modelo sobre el dataset.

Average_precision_Score: Esta es la precisión que se calcula bajo la curva de precisión-exhaustividad realizada para tareas de clasificación binaria o etiquetas múltiples y se puede definir más básicamente como la precisión promedio de las puntuaciones de predicción (F1_score).

En nuestro caso en este problema los datos no están balanceados siendo así la distribución entre valores de la variable objetivo mucho mayo para uno de los dos valores (con una distribución de aproximadamente 80/20), por lo que para analizar **nuestro modelo será más útil observar las métricas de f1_score y average_precision_Score** debido a lo comentado anteriormente. Como se ha podido ver en algunos resultados del apartado b sobre la base de datos breast_cancer, hemos observado que para un número menor de muestras a clasificar de un parámetro que de otro se obtenía una menor precisión en la clasificación de estas mientras que se balanceaba la accuracy_score gracias a la correcta clasificación de los elementos con más muestras en el subconjunto. Por lo que para realizar un correcto análisis deberemos tener en cuenta estas métricas y ver más información en el classification report.

Classification Report:

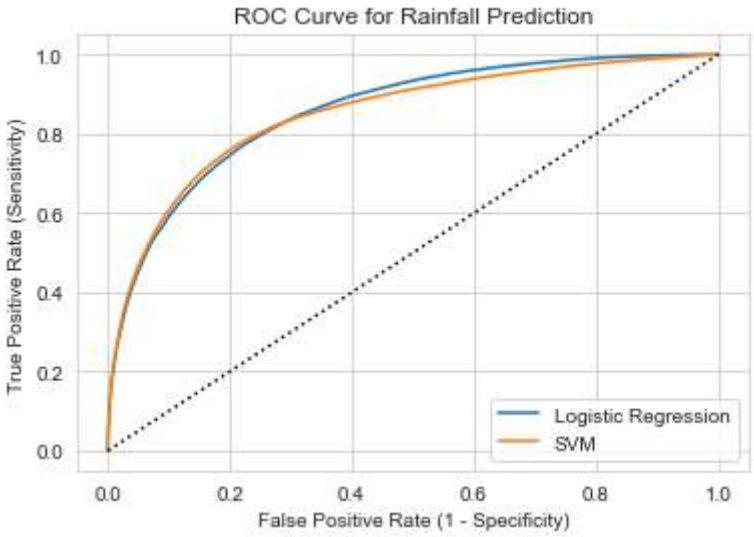
En este informe de clasificación podemos ver los porcentajes de éxito sobre la predicción de los conjuntos de valores que tenemos (train y test) para así evaluar cuál ha sido la precisión para ambos conjuntos en general, mientras que también podemos ver datos más individualizados separados por la clase a clasificar (0 o1 en nuestro caso ya que es un problema de clasificación binaria) Para cada valor podemos obtener las métricas de precisión sobre los porcentajes de acierto del modelo respecto a una clase, f1_score, que no es más que una medida de la precisión del modelo sobre el dataset entrenado en esa clase, recall, que viene a ser la tasa entre verdaderos positivos y falsos positivos y el número de atributos analizado. Por otra parte nos muestra la mediana de los resultados en cada uno de los apartados sobre las dos clases.

Habiendo realizado ambos modelos con una k-fold-cross calidation con 10 iteraciones adjuntamos ahora los resultados obtenidos en cada uno de los dos:

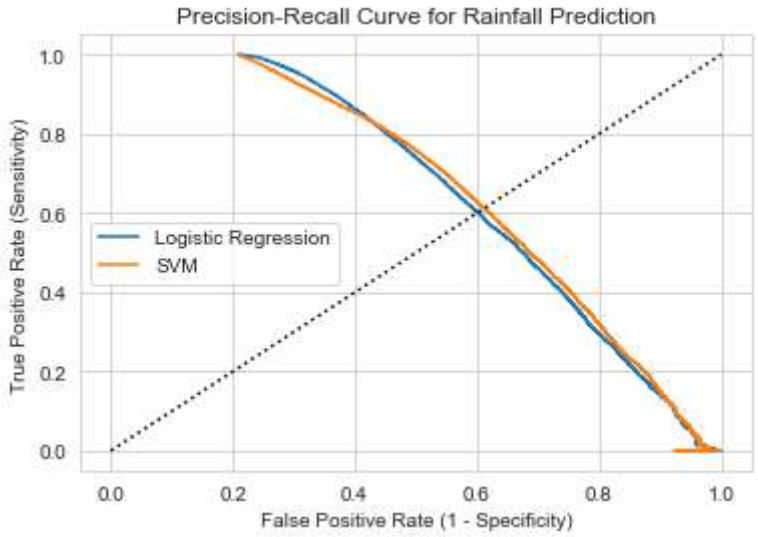
```
Accuracy Regresión Loística:  87.16129519208997 %
F1 score Regresión Logística : 57.229067136032185 %

Classification Report Regresión Logística:
              precision    recall  f1-score   support

         0.0       0.95      0.90      0.92     11513
         1.0       0.51      0.66      0.57      1736

   micro avg       0.87      0.87      0.87     13249
   macro avg       0.73      0.78      0.75     13249
weighted avg       0.89      0.87      0.88     13249

Accuracy SVC:  87.13865197373387 %
F1 score SVC : 50.058616647127785 %

Classification Report SVC:
              precision    recall  f1-score   support

         0.0       0.97      0.89      0.93     12078
         1.0       0.38      0.73      0.50      1171

   micro avg       0.87      0.87      0.87     13249
   macro avg       0.68      0.81      0.71     13249
weighted avg       0.92      0.87      0.89     13249
```

Como podemos ver, nuestros modelos tienen una alta precisión en cuanto al acierto, pero analizando de forma mejor los datos podemos ver que la distribución no es balanceada como habíamos dicho antes. Por lo que el hecho de tener una proporción tan grande de la clase 0 (No llueve mañana), respecto a 1 (si llueve mañana), hace que los resultados que tienen en cuenta el ratio de verdaderos positivos con falsos positivos sea peor para ambos modelos. Sí que es cierto que para la regresión logística este valor es algo más elevado y por la tanto más óptimo, pero no hay demasiadas diferencias significativas. Para corregir esto deberíamos introducir más muestras para entrenar el modelo de la clase 1 para poder estudiar así mejor el comportamiento.

Una vez completado y analizado sobre parámetros ambos modelos realizados, **se procede a realizar las curvas de precisión-sensibilidad y la curva ROC.**



Para interpretar este grafico se debe tener en cuenta que depende de la proporción de verdaderos positivos contra falsos positivos según se varia el umbral de discriminación, el cual es un calor a partir del cual se decide si un caos es positivo. Mientras más alta sea el área entre la curva y el eje de coordenadas X mejor será nuestro modelo dando predicciones, en este caso, podemos observar que no es un modelo es un buen predictor para la lluvia en Australia pero no podría ser todavía mejor.



Por otra parte la curva precisión-recall depende de la relación de los verdaderos positivos con la división de los verdaderos positivos entre la suma de los verdaderos positivos más los falsos negativos. Para interpretar que nuestro modelo es bueno prediciendo los datos este grafico debería ser prácticamente horizontal en el punto más elevado del eje y, hasta el final del eje de las x. Como podemos ver en un principio nuestros modelos no son malos clasificadores pero al aumentar la sensibilidad la precisión decae mucho lo que nos indica ciertos errores en el ratio de los falsos negativos, lo cual es un aspecto claramente a mejorar y que se podría realizar mediante la inclusión de elementos de la clase 1 en el estudio y que la proporción sea distribuida uniformemente.

HYPERPARAMETER SEARCH

Para encontrar los mejores hiperparametros se deben dividir los datos disponibles en subconjuntos de entrenamientos y de test, repitiendo el ciclo de optimización en un número fijo de iteraciones, o hasta que se cumpla una condición, y por último se debe comparar todos los valores de las métricas y así se puede escoger el conjunto de hiperparametros que produzca el mejor valor de las métricas. Para realizar esto **se debe tener una alta capacidad de computo** ya que depende de un número elevado de iteraciones, y la complejidad de un problema multivariable es elevada para ser procesada ya que es igual a número de muestras elevado al cuadrado multiplicado por el número de atributos, y realizar esto un número limitado de veces hasta encontrar la mejor opción es muy costoso computacionalmente.

En el caso de disponer de recursos limitados, la mejor opción será escoger un algoritmo lineal o una regresión logística, debido a que la complejidad de estas opciones es menor, y así se pueden conseguir resultados más rápidos con un índice de acierto similar aunque no será la mejor opción.

Existen varias maneras de encontrar hiperparametros con la librería scikit-optimize de Python la cual incluye varios algoritmos para estimar estos parámetros en función de la naturaleza del problema, entre ellos encontramos **BayesSearchCSV** el cual es capaz de buscar parámetros en modelos de machine learning que resultan en los mejores para el rendimiento de cross validation.

CONCLUSIONES

Después de haber realizado el estudio sobre modelos de clasificación y haber realizado un caso completo sobre estos, podemos concluir que no existe un modelo superior a otro, estos se deben decidir según la naturaleza del problema, los coeficientes, parámetros a aplicar, variables o funciones que puedan complementar a los algoritmos y la regulación.

Por otra parte cabe destacar la importancia de las funciones kernel y las variables slack sobre las máquinas de vectores de soporte, los cuales nos son muy útiles para representar los datos en una dimensión superior de manera rápida y eficiente sin necesidad de calcular las coordenadas en un espacio mayor a la vez que con las variables slack podemos obtener un clasificador más robusto a costa de un margen más amplio para los errores entre los límites de decisión.

Por otra parte en cuanto al procesado de datos, destaca la importancia de la normalización de valores, de manera que esta afecta totalmente a los resultados y a la eficiencia de los algoritmos de clasificación, haciendo así que los datos estén comprendidos en un mismo rango de valores y facilitando la correlación entre sí. De igual forma destaca la transformación de variables categóricas, ya que estas son más difíciles de cuantifica, pero con una buena codificación pueden resultar útiles de cara al posterior estudio y pueden indicar relevancia con la variable objetivo.

Por último y no menos importante, se debe tener en cuenta el número de elementos y atributos a procesar por un clasificador, ya que sobretodo en caso de n o ser lineal la complejidad de procesamiento de estos es elevada, y por tanto requerirá un uso mayor de recursos de computo así como de tiempo de ejecución, lo cual decelera el proceso de estudio y puede resultar tedioso para ajustar ciertos valores así como la búsqueda de hiperparametros.