Pontificia Universidad Javeriana

Taller SVMs

Tecnologías Emergentes

Presentado por:

Nicolás Rincón Ballesteros

Alejandro Suarez Acosta

Andrés García Montoya

Entregado a:

Randy Lancheros

Bogotá D.C

Sábado 13 de abril 2024

Análisis Exploratorio de los Datos

Se realiza un análisis exploratorio de los datos para identificar la relación entre las variables y la variable objetivo, en este caso se elabora un mapa de calor para identificar la correlación entre las variables, el resultado encontrado es el siguiente:

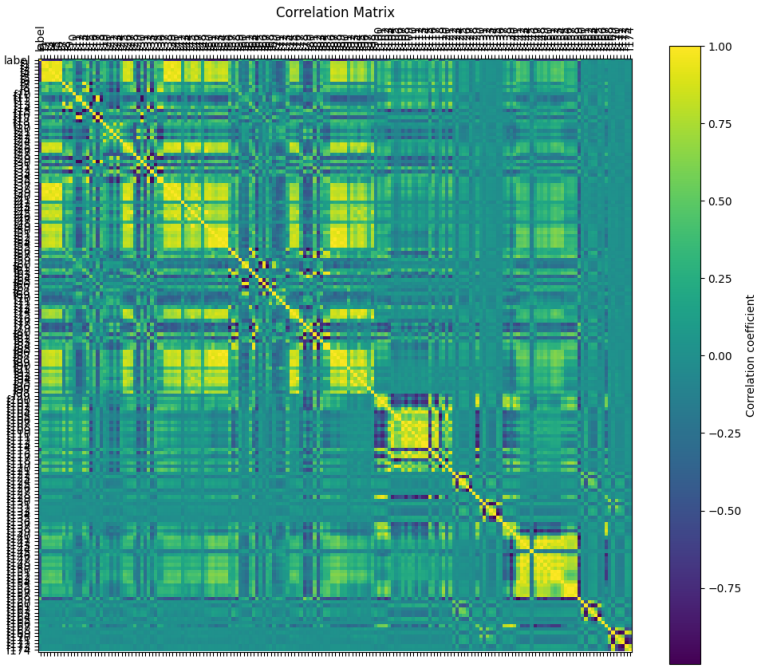

Figura 1. Mapa de calor dataframe. Fuente: Elaboración propia.

Como se puede ver, hay algunas variables que tienen un alto índice de correlación, que en este caso consideraremos que alto es mayor al 95% (0.95). Entonces, se decide realizar un preprocesamiento de los datos, eliminando las variables altamente correlacionadas y haciendo un escalamiento de datos.

Primero, con respecto a la eliminación de las variables, una vez se hace esto, obtenemos la siguiente matriz de correlación:

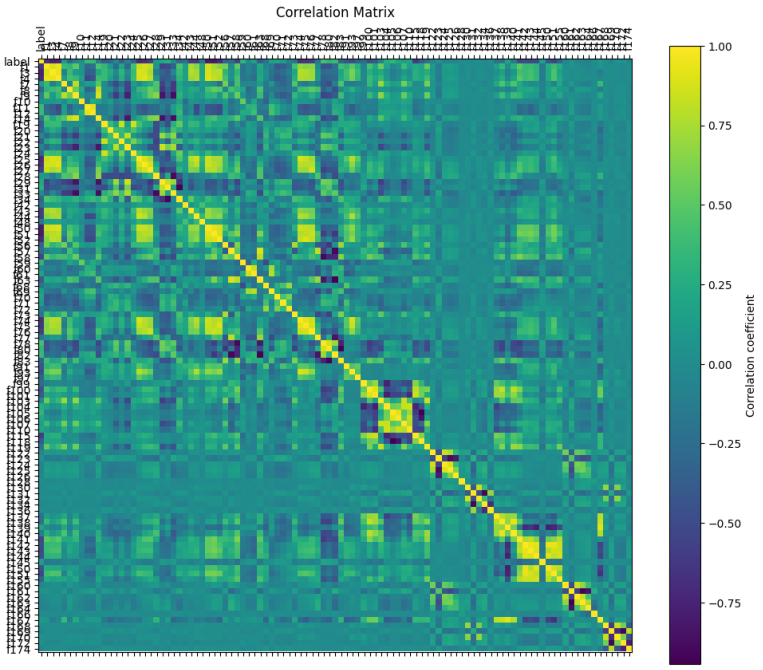

Figura 2. Matriz de correlación tras eliminación variables. Fuente: Elaboración propia.

Ahora, se debe hacer un escalado de datos para que todas las características tengan la misma importancia durante el entrenamiento del modelo, algunas razones para hacerlo es que facilita la convergencia, evita la dominancia de características si algunas de estas tienen mucha mayor magnitud que otras, ignorando otras características y finalmente para que la interpretación de los coeficientes sea más sencilla. Para esto, se procede a hacer una estandarización usando StandardScaler de scikit-learn para que cada característica sea cero y la desviación estándar sea uno.

Obteniendo así el siguiente resultado:

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(dataset.drop('label', axis=1))
scaled_data = pd.DataFrame(scaled_data, columns=dataset.columns[1::])
scaled_data.head()
```

Python

	f1	f3	f4	f7	f8	f9	f10	f11	f12	f14	...	f162	f163	f164	f166	f167
0	0.452194	1.222832	0.901446	-1.433501	0.859635	-0.687087	0.350325	-0.516842	-0.367857	-0.273562	...	0.713910	1.135903	1.395822	0.898027	2.461464
1	0.668218	1.530373	1.202123	-1.581594	1.120757	-0.654580	0.662312	-0.709760	-0.434976	-0.174124	...	0.190167	1.135903	0.258723	-1.847469	-0.046765
2	0.774089	1.629827	1.345995	-1.553474	1.171669	-0.600352	1.033551	-0.888007	-0.464799	-0.115194	...	-0.071690	0.456821	0.542952	-0.172181	0.039718
3	0.700464	1.643904	1.412813	-1.734502	1.696652	-0.436719	1.392446	-1.204425	-0.634058	0.182583	...	-0.071690	0.456821	1.503238	0.245278	1.510077
4	0.701320	1.708013	1.478013	-1.857209	1.902376	-0.416147	1.673608	-1.283512	-0.602220	0.263599	...	0.347323	0.909556	1.218825	0.047318	1.423595

5 rows × 102 columns

Figura 3. Resultados escalado de datos con estandarización. Fuente: Elaboración propia.

Resultados Obtenidos

Después de hacer el preprocesamiento de datos, ya es posible hacer la separación de datos de entrenamiento y prueba para poder usarlos en el entrenamiento de SVMs con distintos

Kernel. En este caso, se dividieron los datos preprocesados se dividieron 75% para el entrenamiento y 25% para la prueba.

- **Kernel Polinomial (grado 2)**

Para entrenar el SVM utilizando el Kernel polinomial de grado 2, se utiliza la función SVC de la siguiente manera:

```
classifier_poly2 = SVC(kernel='poly', degree=2)
classifier_poly2.fit(X_train, y_train)
```

Figura 4. Kernel polinomial grado 2. Fuente: Elaboración propia.

Posteriormente se hace una predicción de datos y se comparan los actuales con los predichos por el SVM.

	Predicted	Actual
100905	3	3
178208	4	4
7105	1	1
277408	6	6
9835	1	1
...	...	...
41129	2	2
4488	1	1
34437	1	1
131598	4	4
100491	3	3

Figura 5. Tabla dato predicho contra actual Kernel polinomial grado 2. Fuente: Elaboración propia.

En la foto anterior, se puede evidenciar que, en los casos mostrados, los valores predichos son iguales a los actuales del dataset. Posteriormente, se realiza la siguiente matriz de confusión para una mejor visualización de los resultados con los valores predichos y actuales de los datos para encontrar cuantas instancias fueron bien o mal clasificadas.

```
array([[ 9828,     0,     4,    13,     2,     5,     3],
       [    1,   898,     0,     0,     0,     0,     0],
       [    2,     1, 18856,     4,     6,    13,     0],
       [   16,     0,     3, 18682,    17,     7,     0],
       [    5,     0,    10,    10, 11544,    99,     0],
       [    8,     0,     6,    21,    70, 21037,     2],
       [    3,     0,     0,     1,     2,     3,   277]])
```

Figura 6. Matriz de confusión Kernel polinómico grado 2. Fuente: Elaboración propia.

Como se ve, gran parte de los datos están clasificados correctamente, estos los representan en la diagonal cada uno de los 7 valores que puede escoger el modelo. Entonces, los datos fuera de la diagonal principal son los errores de clasificación, lo que representa el número

de veces que una clase especifica se clasificó incorrectamente como otra clase. Cada fila de la matriz representa la clase verdadera y cada columna representa la clase predicha.

Una vez se realizó la matriz de confusión, se evalúa la precisión del modelo el cual nos da lo siguiente:

```
accuracy_poly2 = accuracy_score(y_test, y_pred_poly2)
accuracy_poly2
```

```
0.9958629494592371
```

Figura 7. Precisión del modelo Kernel polinómico grado 2. Fuente: Elaboración propia.

Como se evidencia, nos dio una precisión de 0.958629494592371, bastante alta.

- **Kernel Polinomial (grado 3)**

Para entrenar el SVM utilizando el Kernel polinomial de grado 3, se utiliza la función SVC de la siguiente manera:

```
classifier_poly3 = SVC(kernel='poly', degree=3)
classifier_poly3.fit(X_train, y_train)
```

Figura 8. Kernel polinomial grado 3. Fuente: Elaboración propia.

Posteriormente se hace una predicción de datos y se comparan los actuales con los predichos por el SVM.

	Predicted	Actual
100905	3	3
178208	4	4
7105	1	1
277408	6	6
9835	1	1
...	...	...
41129	2	2
4488	1	1
34437	1	1
131598	4	4
100491	3	3

Figura 9. Tabla dato predicho contra actual Kernel polinomial grado 3. Fuente: Elaboración propia.

En la foto anterior, se puede evidenciar que, en los casos mostrados, los valores predichos son iguales a los actuales del dataset. Posteriormente, se realiza la siguiente matriz de confusión para una mejor visualización de los resultados con los valores predichos y actuales de los datos para encontrar cuantas instancias fueron bien o mal clasificadas.

```
array([[ 9832,     0,     2,    12,     2,     4,     3],
       [    1,   898,     0,     0,     0,     0,     0],
       [    3,     2, 18860,     7,     2,     8,     0],
       [   11,     0,     5, 18680,    19,    10,     0],
       [    3,     1,     9,    13, 11567,    75,     0],
       [    5,     0,     6,    15,    57, 21058,     3],
       [    4,     0,     3,     0,     4,     1,   274]])
```

Figura 10. Matriz de confusión Kernel polinomial grado 3. Fuente: Elaboración propia.

Como se ve, gran parte de los datos están clasificados correctamente, estos los representan en la diagonal cada uno de los 7 valores que puede escoger el modelo. Entonces, los datos fuera de la diagonal principal son los errores de clasificación, lo que representa el número de veces que una clase especifica se clasificó incorrectamente como otra clase. Cada fila de la matriz representa la clase verdadera y cada columna representa la clase predicha.

Una vez se realizó la matriz de confusión, se evalúa la precisión del modelo el cual nos da lo siguiente:

```
accuracy_poly3 = accuracy_score(y_test, y_pred_poly3)
accuracy_poly3
```

```
0.9964399268343583
```

Figura 11. Precisión del modelo Kernel polinómico grado 3. Fuente: Elaboración propia.

Como se ve, nos dio una precisión de 0.9964399268343583 bastante alta.

- **Kernel RBF**

Para entrenar el SVM utilizando el Kernel RBF, se utiliza la función SVC de la siguiente manera

```
classifier_rbf = SVC(kernel = 'rbf', random_state = 0)
classifier_rbf.fit(X_train, y_train)
```

Figura 12. Kernel rbf. Fuente: Elaboración propia.

Luego se visualizan algunos de los valores predichos y actuales del modelo

	Predicted	Actual
100905	3	3
178208	4	4
7105	1	1
277408	6	6
9835	1	1
...	...	...
41129	2	2
4488	1	1
34437	1	1
131598	4	4
100491	3	3

Figura 13. Tabla dato predicho contra actual Kernel rbf. Fuente: Elaboración propia.

Entonces, se realiza la siguiente matriz de confusión para una mejor visualización de los resultados con los valores predichos y actuales de los datos para encontrar cuantas instancias fueron bien o mal clasificadas.

```
array([[ 9833,     0,     1,     6,     4,     3,     8],
       [    0,   893,     0,     0,     0,     0,     6],
       [    1,     2, 18861,     3,     3,    10,     2],
       [    6,     1,     6, 18692,    11,     9,     0],
       [    5,     0,     5,    15, 11568,    74,     1],
       [    0,     0,     6,    12,    46, 21076,     4],
       [    1,     0,     0,     0,     1,     2,   282]])
```

Figura 14. Matriz de confusión Kernel rbf. Fuente: Elaboración propia.

Como se puede ver, gran parte de los datos están clasificados correctamente, estos están representados en la diagonal por cada uno de los 7 valores que puede escoger el modelo. Entonces, los datos que están fuera de la diagonal principal son los errores de clasificación, esto representa la cantidad de veces que una clase especifica fue clasificada incorrectamente como otra clase. Cada fila de la matriz representa la clase verdadera y cada columna representa la clase predicha.

Una vez se realizó la matriz de confusión, se evalúa la precisión del modelo el cual nos da lo siguiente:

```
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
accuracy_rbf
```

```
0.9968818669514725
```

Figura 15. Precisión del modelo Kernel rbf. Fuente: Elaboración propia.

Como se logra ver, nos dio una precisión de 0.9968818669514725 la cual es bastante alta.

Conclusiones

Durante este taller, se ha realizado una exhaustiva exploración sobre el empleo de las máquinas de soporte vectorial (SVM) mediante la aplicación de diversos kernels en el análisis de datos. Los resultados obtenidos han evidenciado la capacidad y versatilidad de los SVM para llevar a cabo una clasificación precisa y eficiente de datos complejos.

La eliminación de variables altamente correlacionadas demostró ser una estrategia efectiva para simplificar el modelo sin sacrificar el rendimiento. Además, la estandarización de las características permitió que el modelo no estuviera sesgado hacia variables con mayor magnitud, facilitando un aprendizaje más equitativo entre las características.

Es notable que la elección del kernel tiene un impacto significativo en la efectividad del modelo. Específicamente, los kernels polinomial y RBF han demostrado ser particularmente efectivos, alcanzando precisiones superiores al 99%. Esto subraya la importancia de evaluar minuciosamente la naturaleza de los datos y los objetivos específicos del análisis para seleccionar el kernel más adecuado.

A pesar de que las diferencias en el porcentaje de predicciones correctas entre los tres kernels no fueron significativas, se pudo observar que el kernel RBF obtuvo el puntaje más alto, con una precisión de 0.9968818669514725, donde un valor cercano a 1 representa una alta exactitud en las predicciones. Le siguen el kernel polinómico de grado 3, con un puntaje de 0.9964399268343583, y finalmente el kernel polinómico de grado 2, con un puntaje de 0.958629494592371.