Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Estado de México

“Apegándome a la Integridad Académica de los Estudiantes del Tecnológico de Monterrey, me comprometo a que mi actuación en esta actividad esté regida por la integridad académica. En congruencia con el compromiso adquirido, realizaré este trabajo de forma honesta y personal, para reflejar, a través de él, mi conocimiento y aceptar, posteriormente, la evaluación obtenida”

Inteligencia Artificial Avanzada para la ciencia de datos

Momento de retroalimentación

Integrantes:

A01379571 | Juan Daniel Aranda Morales | IRS | a01379571@itesm.mx

Profesor:

Jorge Adolfo Ramírez Uresti

Fecha: 13/09/2022

Introducción

Para esta actividad se requiere el uso de una librería especializada en aprendizaje máquina para generar un modelo que pueda realizar predicciones exactas con base en algún dataset. Para esto se utilizará la base de datos del Titanic original que se encuentra en Kaggle. Los datos se encuentran divididos en Train y Test. El programa debe funcionar independientemente de cualquier IDE o "Notebook".

Desarrollo

Primeramente se realizó un pre-procesamiento de los datos, en donde se reemplazaron las variables categóricas por variables numéricas. Se rellenaron los datos faltantes con la media como es el caso de la edad y se eliminaron las variables que no eran relevantes como el nombre, ID, etc.

Se entrenaron tres modelos diferentes.

Regresión logística

```
#Regresion logística
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print("Precision con regresión logística: ")
print(logreg.score(X_train, y_train))
```

Máquina de soporte vectorial

```
#Maquina de soporte vectorial
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print("Precision MSV: ")
print(svc.score(X_train, y_train))
```

Knn neighbors

```
#K neighbors
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Precision Knn: ")
print(knn.score(X_train, y_train))
```

Se obtuvieron los siguientes resultados:

```
Precision con regresión logística:
0.8102893890675241
Precision MSV:
0.6897106109324759
Precision Knn:
0.882636655948553
```

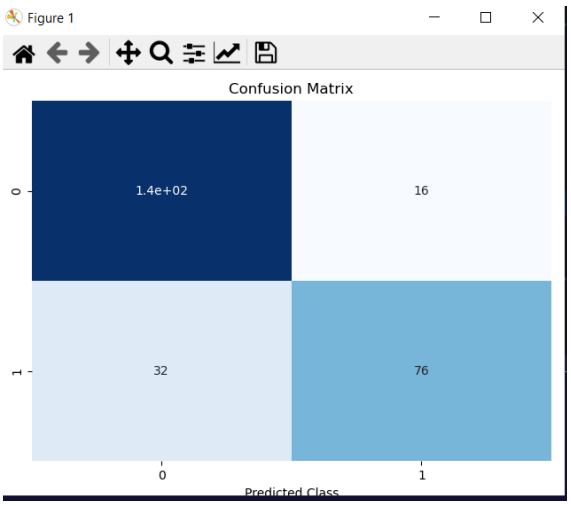
Como podemos observar el modelo con mejor exactitud es el Knn, debido a que tiene una accuracy de .88. Al ejecutar el algoritmo variando el hiper parámetro de los vecinos en el algoritmo Knn poniéndole un valor de 5, nos da el siguiente resultado.

Para encontrar el modelo más eficiente se realizaron 3 iteraciones y obtuvimos los siguientes valores para el accuracy:

Modelo	Promedio	Desviación estándar
Regresión logística	0.7902	0.0
MSV	0.6367	0.0
KNN	0.7378	1.11e-16

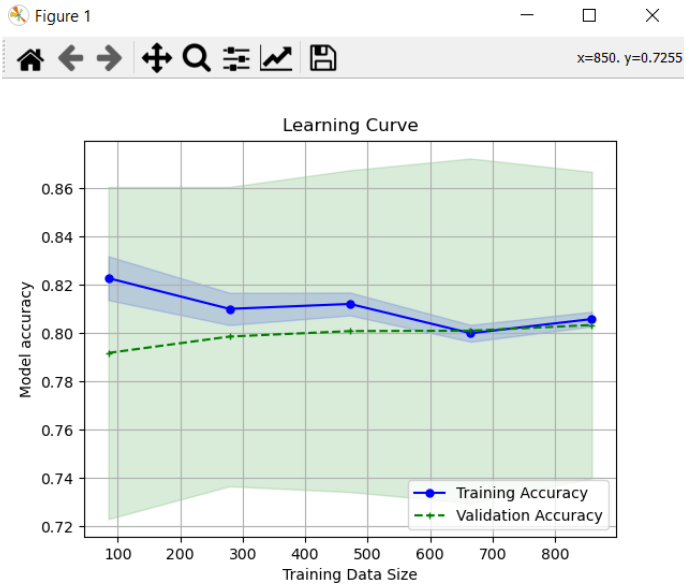
Como podemos observar el modelo más exacto es el de regresión logística, al tener una desviación estándar de 0, podemos esperar un rendimiento estable del modelo, esto quiere decir que el puntaje del accuracy no tiene variaciones significativas en cada iteración de prueba del modelo.

Matriz de confusión regresión logística



Como podemos observar los verdaderos positivos y verdaderos negativos son mayores que los falsos por lo que el modelo cumple con la precisión dada anteriormente.

Curva de aprendizaje regresión logística



Como podemos observar, la varianza y el baias se encuentran en un buen rango por lo que el modelo de regresión logística no muestra Overfitting o Underfitting, por lo que nuestro modelo está generalizando el problema de manera eficiente.

Optimización del modelo

Utilizando el método de Grid Search se optimizó el modelo para obtener una exactitud mayor a las anteriores.

```
grid={"C":np.logspace(-3,3,7), "penalty":["l1","l2"]}
logreg=LogisticRegression()
logreg_cv=GridSearchCV(logreg,grid,cv=10)
logreg_cv.fit(X_train,y_train)
```

```
score 0.8014981273408239
```

Conclusión

En conclusión, la eficiencia de nuestro modelo va a depender mucho del procesado que le hagamos a nuestros datos, pero además de esto debemos tomar en cuenta probar diferentes algoritmos de ML ya que alguno nos puede funcionar mejor en determinado caso, además se debe tomar en cuenta la variación de los hiper parámetros ya que el aumentarlos no quiere decir que tendremos un mejor resultado, cada situación es diferente por lo que se deben hacer diferentes pruebas o iteraciones variando los hiper parámetros. Para la optimización de los parámetros se pueden usar métodos como grid search y de esta forma obtener un modelo con un mejor desempeño.