Inteligencia Artificial Avanzada para la Ciencia de Datos (TC3006C)

Grupo 102

Análisis y Reporte sobre el desempeño del modelo

Profesores:

César Javier Guerra Páramo

Andrea Piñeiro Cavazos

A01705681

Campus Querétaro,

Domingo 11 de septiembre de 2022

Análisis y Reporte sobre el desempeño del modelo

La implementación elegida para el análisis fue la de aprendizaje máquina utilizando un Framework. Encontrada dentro del mismo repositorio en el documento "car_purchase_tensorflow.py".

Ahí mismo se encuentra el primer modelo (desarrollado para la entrega) y el segundo modelo mejorado.

Datos

Los datos seleccionados fueron los sigueintes: https://www.kaggle.com/datasets/gabrielsantello/cars-purchase-decision-dataset. Obtenido de la página de Kaggle.

Este dataset busca predecir si cierto cliente comprará o no un carro dados los siguientes datos:

- Género
- Edad
- Salario Anual

No cuenta con división entre entrenamiento y prueba.

Separación de datos (Train/Test/Validation)

Para separar los datos en **entrenamiento** y **prueba** se usó sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html). Que nos ayuda a dividir los arreglos o matrices en sets para entrenamiento y prueba aleatorios.

Como se puede observar yo especifiqué que se use el 20% de los datos para test. Y de igual forma especifiqué una semilla para que los datos se dividan siempre igual y al correrlo varias veces siempre obtengamos el mismo resultado.

```
# Dividir en train y test
x_train, x_test, y_train, y_test = train_test_split(X_trans, Y, test_size = 0.2, random_state = 55)
```

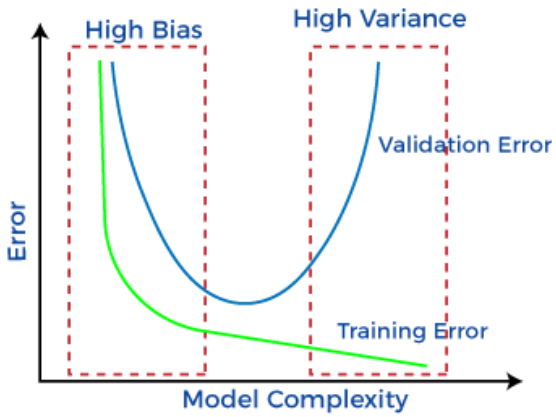
Para el **set de validación**, use la librería de tf.keras que cuenta con la función de fit para entrenar el modelo. (https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit)

Esta función cuenta con el argumento de **set de validación**, el cual recibe el porcentaje de datos dentro del set de entrenamiento que queremos tomar como validación y evalúa la pérdida y otras métricas.

```
# Entrenar el modelo
print('Inicio del entrenamiento')
historia = model2.fit(x_train, y_train, epochs = 300, verbose = True, validation_split = 0.2)
print("Modelo entrenado")
```

Modelo 1

Diagnóstico y explicación del grado de sesgo

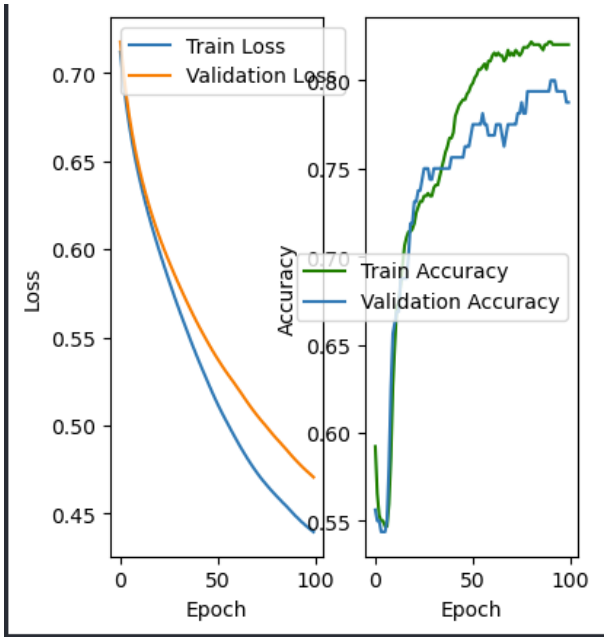


Como se puede observar, tenemos un Sesgo alto cuando el error tanto en el Entrenamiento como el set de Validación es alto. Pues significa que no estamos obteniendo el resultado esperado en ninguno de los sets.

```
Epoch 100/100
20/20 [==============================] - 0s 2ms/step - loss: 0.4396 - binary_accuracy: 0.8203 - val_loss: 0.4706 - val_binary_accuracy: 0.7875
```

SESGO: MEDIO

Como podemos observar para **training** tenemos un accuracy de 82% y para **validation** tenemos uno de 78%. Lo que significa que para ambos tenemos un error medio, no es un error tan malo pero no es un buen accuracy.



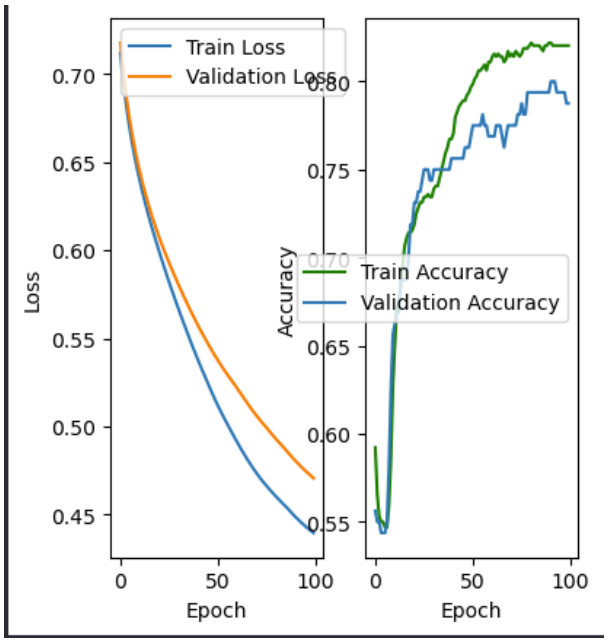
Diagnóstico y explicación del grado de la varianza

Tenemos varianza alta cuando hay poco error en el **training set** y un alto error para el **validation set**, lo cual significaría que el modelo tiene sobre ajuste y se memorizó los datos

```
Epoch 100/100
20/20 [==============================] - 0s 2ms/step - loss: 0.4396 - binary_accuracy: 0.8203 - val_loss: 0.4706 - val_binary_accuracy: 0.7875
```

VARIANZA: BAJA

Como mencionado anteriormente para **training** tenemos un accuracy de 82% y para validation tenemos uno de 78%, la diferencia es de un 4% por lo que no hay gran diferencia entre los resultados, lo que significa que tenemos varianza baja.



Diagnóstico y explicación del nivel de ajuste del modelo

Nivel: Underffit

Como se puede observar en la gráfica anterior, nuestra gráfica de pérdida baja de manera rápida tanto para train como para validation. Sin embargo, no alcanza a llegar a una derivada cerca de cero, por lo que no se acerca a ser horizontal al final. Por lo que la modelo esta subajustado; falta más epochs para poder entrenar al modelo más y lograr acercarnos a una derivada 0.

Modelo 2

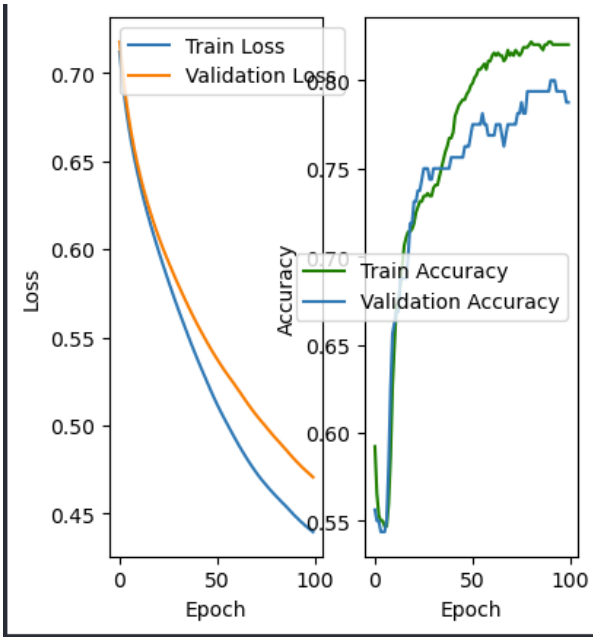
Para mejorar el modelo se agregaron 2 capas ocultas más:

- Para la primera capa se usa la función de activación *relu*
- Para la segunda capa se usa la función de activación *relu*
- Para la tercera capa se usa la función de activación *relu*
- Para la cuarta capa se usa la función de activación *sigmoid*

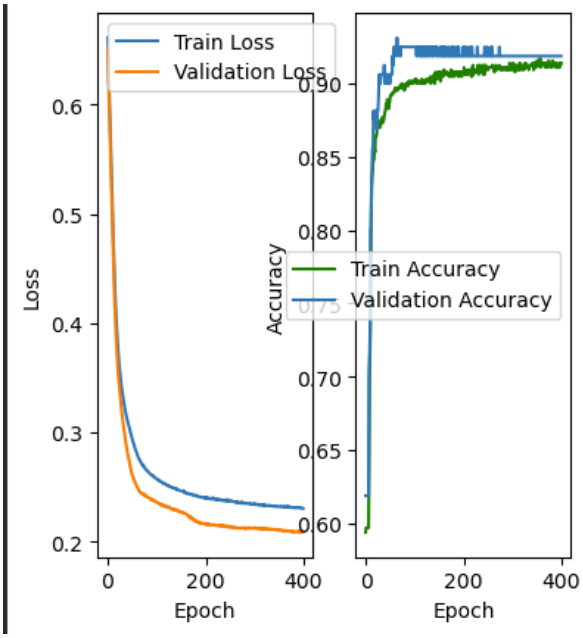
El modelo usa el algoritmo *adam* de optimización y las métricas son *BinaryCrossentropy* y *BinaryAccuracy*, igual que en el modelo anterior para una mejor comparación.

De igual manera se incrementó el número de epochs de 150 a 400

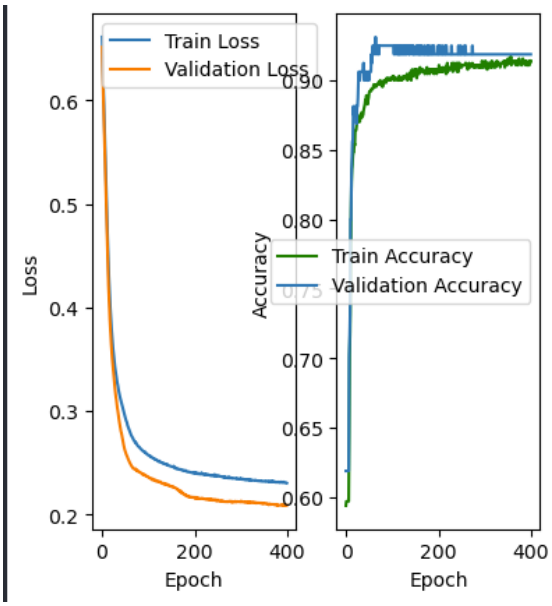
Modelo 1



Modelo 2



Diagnóstico y explicación del grado de sesgo

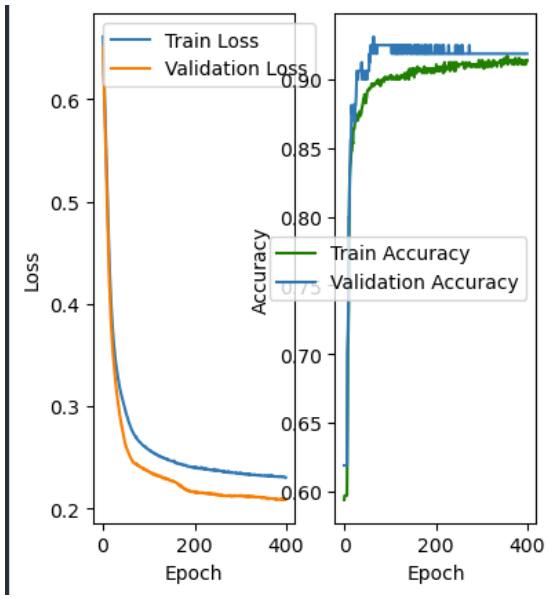


Como se puede observar, tenemos un Sesgo alto cuando tanto el Training como el Validation Error son altos. Pues significa que no estamos obteniendo el resultado esperado en ninguno de los sets.

```
Epoch 400/400
20/20 [==============================] - 0s 3ms/step - loss: 0.2302 - binary_accuracy: 0.9141 - auc_36: 0.9632 - val_loss: 0.2087 - val_binary_accuracy: 0.9187 -
val_auc_36: 0.9638
```

SESGO: Bajo

Como podemos observar para **training** tenemos un accuracy de 91%, para validation tenemos uno de 91%. Lo que significa que para ambos tenemos un error bajo, esta es una muy buena predicción para ambos.



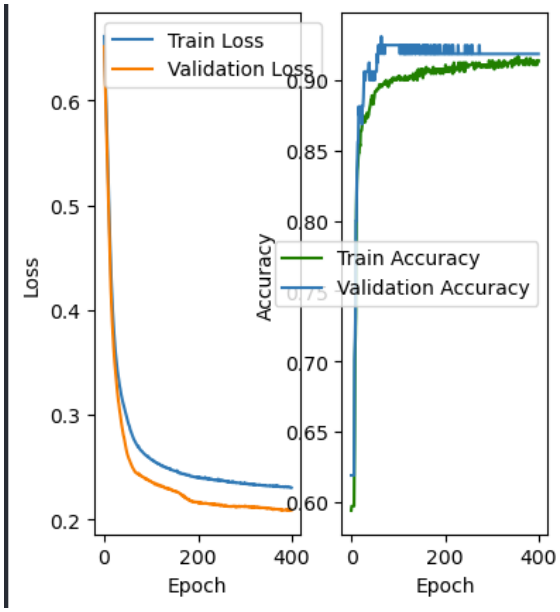
Diagnóstico y explicación del grado de la varianza

Tenemos varianza alta cuando hay poco error en el **training set** y un alto error para el **validation set**, lo cual significaría que el modelo tiene overfitting y se memorizó los datos

```
Epoch 400/400
20/20 [==============================] - 0s 3ms/step - loss: 0.2302 - binary_accuracy: 0.9141 - auc_36: 0.9632 - val_loss: 0.2087 - val_binary_accuracy: 0.9187 -
val_auc_36: 0.9638
```

VARIANZA: BAJA

Como mencionado anteriormente para **training** tenemos un accuracy de 91% y para validation tenemos uno de 91% también, no hay diferencia entre los resultados, lo que significa que tenemos varianza baja.



Diagnóstico y explicación del nivel de ajuste del modelo

Nivel: Fit

Como se puede observar en la gráfica anterior, nuestra gráfica de pérdida baja de manera rápida tanto para train como para validation. Y ahora si alcanza a llegar a una derivada cerca de cero y se acerca a ser horizontal al final. De igual manera para el Accuracy, este llega a ser casi horizontal para ambos y tenemos buenos resultados en los dos. Es por eso que el modelo esta ajustado correctamente. Si tuviéramos mayor error en el **validation** sería por que se aprendió los datos y se sobre ajustó, y si tuviéramos poca **accuracy** sería porque falto entrenar más al modelo y hubo subajuste.

Con esto se concluye que hubo grandes mejoras después de cambiar el modelo y agregarle nuevas capas.