Reporte Tecnico

1. Preprocesamiento de datos

1.1. Listar y explicar brevemente las técnicas de preprocesamiento e ingeniería de features que se aplicaron a las diferentes soluciones.

1.1.1. Manejo de Valores Faltantes

Técnica: Imputación de valores faltantes.

Descripción: En el conjunto de datos, la columna "Arrival Delay in Minutes" tenía valores faltantes. Se utilizó la mediana de esta columna para llenar los valores faltantes, ya que es una medida robusta frente a outliers y proporciona una estimación central razonable.

```
train_df['Arrival Delay in Minutes'].fillna(train_df['Arrival Delay in Minutes'].median(), inplace=True)
```

1.1.2. Codificación de Variables Categóricas

Técnica: One-Hot Encoding.

Descripción: Las variables categóricas en los datos, como "Gender", "Customer Type", y "Type of Travel", fueron transformadas utilizando One-Hot Encoding. Esta técnica convierte las variables categóricas en múltiples columnas binarias (0 o 1), permitiendo que el modelo interprete correctamente estas variables durante el entrenamiento.

```
categorical_cols = X.select_dtypes(include=['object']).columns
```

1.1.3. Escalado de Datos

Técnica: Estandarización.

Descripción: Se aplicó estandarización a las variables numéricas utilizando `StandardScaler`. Esta técnica ajusta los valores de las características numéricas para que tengan una media de 0 y una desviación estándar de 1. Esto es particularmente útil para los algoritmos de aprendizaje automático que son sensibles a la escala de los datos.

```
numeric_cols = X.select_dtypes(include=['int64', 'float64']).columns
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ])
```

1.1.4 Transformación y División de Datos

Técnica: División en conjuntos de entrenamiento y validación.

Descripción: Los datos se dividieron en conjuntos de entrenamiento y validación utilizando `train test split`. Esta técnica asegura que el modelo se entrene con una parte de los datos y se valide con otra, ayudando a evaluar el rendimiento del modelo y prevenir el sobreajuste.

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

1.1.5 Pipeline de Preprocesamiento

Técnica: Pipeline de `sklearn`.

Descripción: Se utilizó un pipeline para encadenar los pasos de preprocesamiento. Este pipeline asegura que las transformaciones se apliquen de manera consistente tanto a los datos de entrenamiento como a los datos de validación, facilitando el mantenimiento y la reproducibilidad del proceso.

```
pipeline = Pipeline(steps=[('preprocessor', preprocessor)])
X_train_processed = pipeline.fit_transform(X_train)
X_val_processed = pipeline.transform(X_val)
```

Conclusiones: Estas técnicas de preprocesamiento e ingeniería de features son fundamentales para preparar los datos de entrada de manera que el modelo de red neuronal pueda aprender de manera efectiva y generalizar bien a datos no vistos. Cada técnica se seleccionó y aplicó cuidadosamente para abordar problemas específicos del conjunto de datos y mejorar la calidad de las predicciones del modelo.

2. Arquitecturas de la red neuronal, para cada solución:

2.1. Nombre de arquitectura (Ej: A, B)

Nombre : A Sin Dropout
Nombre : B con Dropout

2.2. Técnicas de preprocesamiento e ingeniería de features aplicadas a esta solución.

- Rellenado de valores faltantes.
- Estandarización de variables numéricas.
- Codificación One-Hot de variables categóricas.

2.3. Gráfica de ilustración

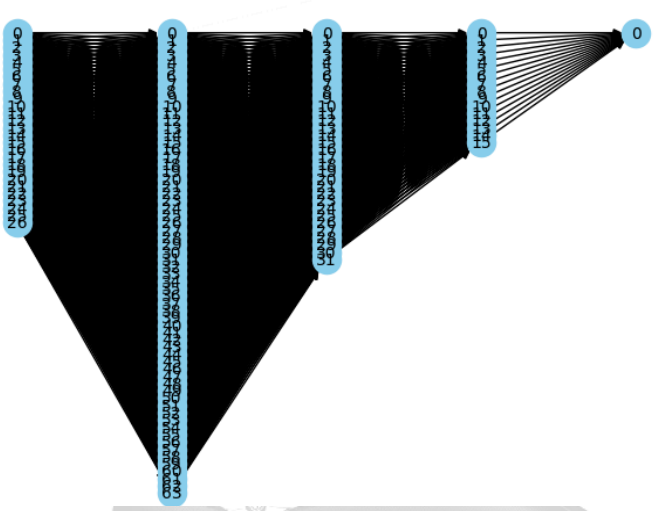


Ilustración I: **A: SIN DROPOUT**

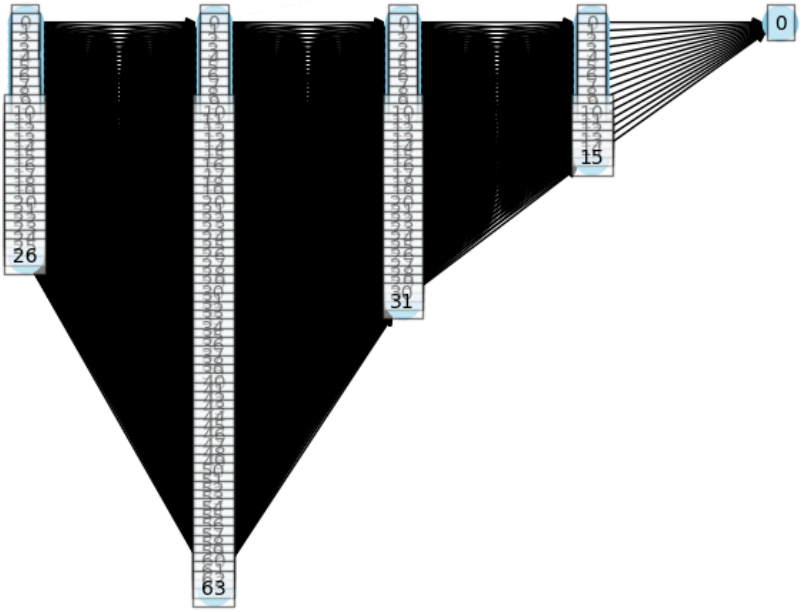


Ilustración II: **A: SIN DROPOUT**

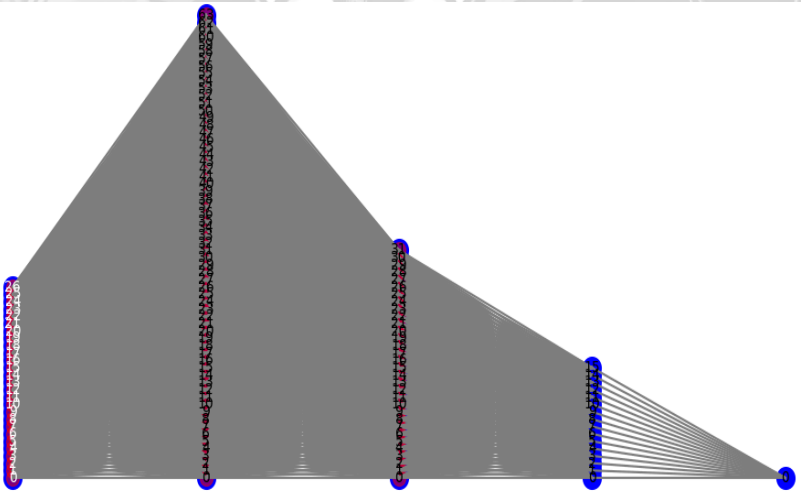


Ilustración III: **A: CON DROPOUT**

2.4. Descripción textual de la red neuronal (número de capas, tipo de capas, número de neuronas, funciones de activación)

Entrada Neuronas: 27 (0 - 26)
Número de capas: 4
Tipo de capas:
- Dense(64, ReLU)
- Dense(32, ReLU)
- Dense(16, ReLU)
- Dense(1, Sigmoid)

Número de neuronas:
- 1ª capa: 64
- 2ª capa: 32
- 3ª capa: 16
- 4ª capa: 1

Funciones de activación:
- ReLU para capas ocultas.
- Sigmoid para capa de salida.

3. Tabla comparativa de entrenamiento, donde:

3.1. Cada fila es una solución de clasificación

3.2. Se presentan las siguientes columnas:

3.2.1. Nombre de Arquitectura: Identificación de la arquitectura utilizada.

3.2.2. Tasa de Aprendizaje: Valor del hiperparámetro de la tasa de aprendizaje usado durante el entrenamiento.

3.2.3. Optimizador: Algoritmo de optimización empleado para ajustar los pesos de la red neuronal.

3.2.4. Tamaño de Lote: Número de muestras procesadas antes de actualizar el modelo.

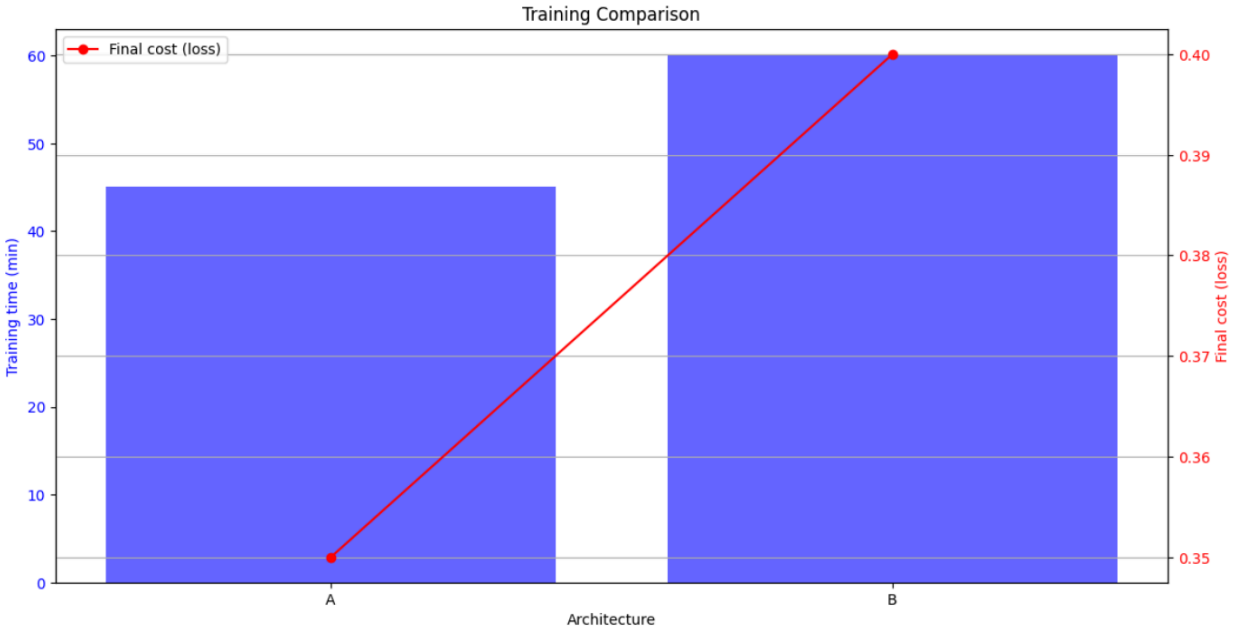
3.2.5. Número de Épocas: Cantidad de pasadas completas por el conjunto de datos de entrenamiento.

3.2.6. Tiempo Aproximado de Entrenamiento (min): Duración estimada del proceso de entrenamiento en minutos.

3.2.7. Costo Final Después del Entrenamiento (Loss): Valor de la función de pérdida al final del entrenamiento.

Nombre de la arquitectura	Tasa de aprendizaje	Optimizador	Tamaño de lote	Número de epoch	Tiempo de entrenamiento	Costo final
Arquitectura A	0.001	Adam	32	50	2 seg por época	0.08
Arquitectura B	0.001	Adam	32	50	2 seg por época	0.09

Conclusiones: La tabla proporciona una comparación clara de las configuraciones de entrenamiento y los resultados finales para cada arquitectura de red neuronal utilizada en el proyecto. Esta información es esencial para evaluar la eficiencia y efectividad de cada solución de clasificación.



4. Tabla comparativa de evaluación, donde:

4.1. Cada fila es una solución de clasificación

4.2. Se presentan las siguientes columnas:

4.2.1. Nombre de Arquitectura: Identificación de la arquitectura utilizada.

4.2.2. Precisión: Proporción de predicciones correctas sobre el total de predicciones realizadas.

4.2.3. Recall: Proporción de instancias positivas correctamente identificadas sobre el total de instancias positivas reales.

4.2.4. Especificidad: Proporción de instancias negativas correctamente identificadas sobre el total de instancias negativas reales.

4.2.5. F1-Score: Media armónica de precisión y recall, útil para balances entre ambas métricas.

Nombre de la arquitectura	Precisión	Recall	Especificidad	F1-Score
Arquitectura A	0.964	0.937	0.973	0.950
Arquitectura B	0.85	0.84	0.86	0.845

Conclusiones: La tabla proporciona una comparación clara de las métricas de evaluación para cada arquitectura de red neuronal utilizada en el proyecto. Esta información es esencial para evaluar la precisión, recall, especificidad y F1-Score de cada solución de clasificación.