


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score

# Step 1: Load the Titanic dataset
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
data = pd.read_csv(url)

# Step 2: Data Preprocessing
features = data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
features['Sex'] = features['Sex'].map({'male': 0, 'female': 1})
features['Age'].fillna(features['Age'].median(), inplace=True)
features['Fare'].fillna(features['Fare'].median(), inplace=True)

target = data['Survived']

# Step 3: Feature scaling
scaler = StandardScaler()
features = scaler.fit_transform(features)
```

 <ipython-input-1-febdfef49189>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
features['Sex'] = features['Sex'].map({'male': 0, 'female': 1}) # Encode 'Sex' as binary
<ipython-input-1-febdfef49189>:19: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the

```
features['Age'].fillna(features['Age'].median(), inplace=True) # Fill missing Age with median
<ipython-input-1-febdfef49189>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
features['Age'].fillna(features['Age'].median(), inplace=True) # Fill missing Age with median
<ipython-input-1-febdfef49189>:20: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the

```
features['Fare'].fillna(features['Fare'].median(), inplace=True) # Fill missing Fare with median
<ipython-input-1-febdfef49189>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
features['Fare'].fillna(features['Fare'].median(), inplace=True) # Fill missing Fare with median
```

```
# Step 4: Define a function to build the DNN model
def build_model():
    model = Sequential()
    model.add(Dense(16, input_dim=features.shape[1], activation='relu'))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(1, activation='sigmoid')) # Output layer for binary classification
    model.compile(loss='binary_crossentropy', optimizer=Adam(), metrics=['accuracy'])
    return model

# Step 5: Set up K-fold cross-validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
fold_no = 1
histories = []
accuracies = []

# Step 6: Train and validate the model using 5-fold cross-validation
for train_index, test_index in kfold.split(features):
    X_train, X_test = features[train_index], features[test_index]
    y_train, y_test = target.iloc[train_index], target.iloc[test_index]


    model = build_model()

    history = model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0, validation_data=(X_test, y_test))
    histories.append(history)

    y_pred = (model.predict(X_test) > 0.5).astype("int32")
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)

    print(f"Fold {fold_no} - Accuracy: {accuracy:.4f}")
    fold_no += 1

# Step 7: Calculate and display the average accuracy across all folds
avg_accuracy = np.mean(accuracies)
print(f"Average Accuracy across all folds: {avg_accuracy:.4f}")
```

 /usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an
super().__init__(activity_regularizer=activity_regularizer, **kwargs)

```
6/6 ━━━━━━━━━━━ 0s 8ms/step
Fold 1 - Accuracy: 0.8212
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
6/6 ━━━━━━━━━━━ 0s 8ms/step
Fold 2 - Accuracy: 0.7978
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
WARNING:tensorflow:5 out of the last 13 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x7c14874c8280> triggered tf.function retracing. Tracing
6/6 ━━━━━━━━━━━ 0s 12ms/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Fold 3 - Accuracy: 0.8876
WARNING:tensorflow:5 out of the last 13 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x7c148636a680> triggered tf.function retracing. Tracing
6/6 ━━━━━━━━━━━ 0s 8ms/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Fold 4 - Accuracy: 0.7865
```

6/6 0s 8ms/step
Fold 5 - Accuracy: 0.8258
Average Accuracy across all folds: 0.8238

```
# Step 8: Plot training and validation accuracy and loss for the last fold
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(histories[-1].history['accuracy'], label='Training Accuracy')
plt.plot(histories[-1].history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy (Last Fold)')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

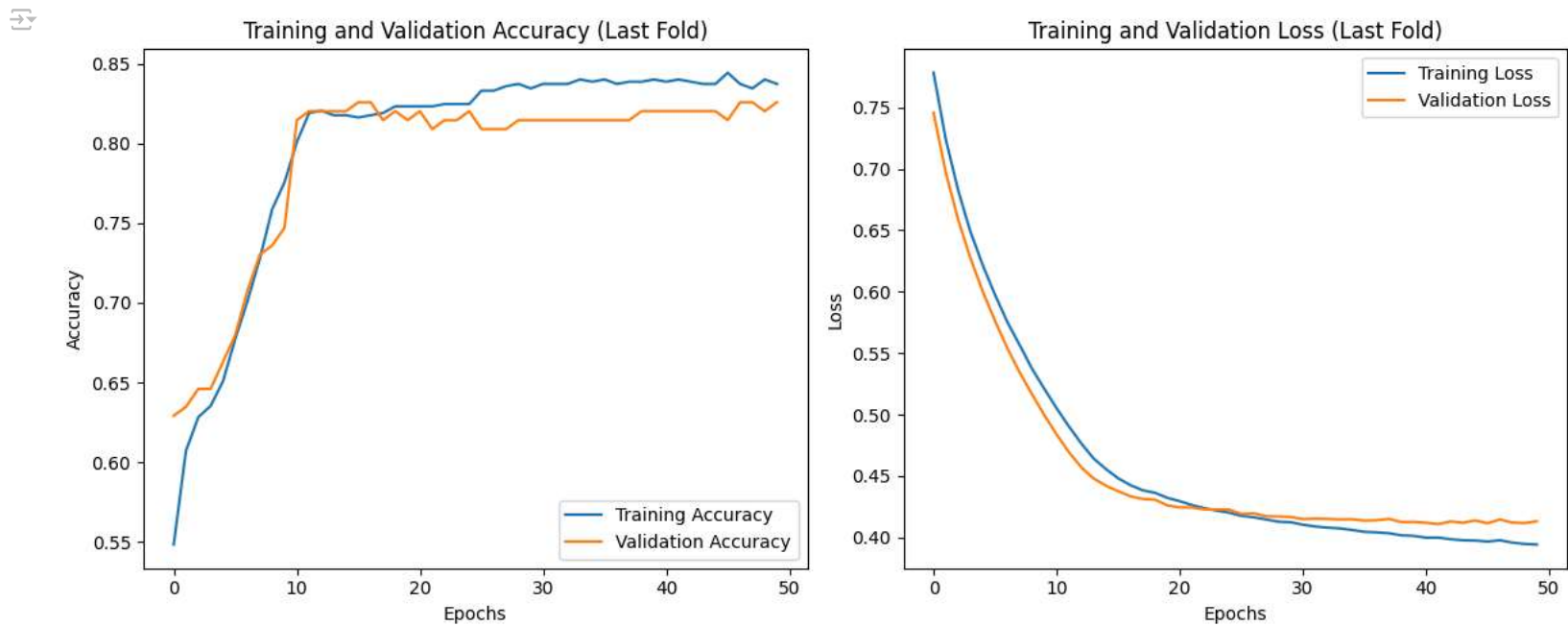
plt.subplot(1, 2, 2)
plt.plot(histories[-1].history['loss'], label='Training Loss')
plt.plot(histories[-1].history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss (Last Fold)')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

model.save('titanic_model_cv.h5')

# Step 10: Make predictions on a sample input from the last fold's model
sample_input = np.array([[1, 1, 25, 0, 0, 100]])
sample_input = scaler.transform(sample_input)
prediction = model.predict(sample_input)
print(f"Sample Prediction (Survival Probability): {prediction[0][0]:.4f}")

print("Predicted Class (0: Did not survive, 1: Survived):", int(prediction[0][0] > 0.5))
```



WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format via `model.save_format('tf')` or `keras.saving.save_model(model, filepath, save_format='tf')`.

1/1 0s 19ms/step
Sample Prediction (Survival Probability): 0.9629
Predicted Class (0: Did not survive, 1: Survived): 1
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
warnings.warn(

Start coding or generate with AI.