Predicting Synaptic Activity Based on Axonal-Dendritic Proximity

Michael Popa (Kaggle Name: mrew01)

ELEC 478 Machine Learning

Dr. Allen Genevera

Rice University, Fall 2023

Public Leadboard - Placed 22nd

Private Leadboard - Placed 4th

The Goal:

When the axon of one neuron comes close to the dendrite of another neuron, we call this an axonal-dendritic proximity, or **ADP**. Every synapse results from an ADP, but not every ADP results in a synapse. When the ADP contains a synapse, we say the ADP has been "converted" to a synapse. It is of major interest to neuroscience what rules determine whether ADP's will convert to a synapse. In other words, what drives neurons to connect to each other, given that are within proximity (i.e. physically close enough)?

The machine learning task is to predict whether an ADP results in a synapse, and is hence a binary classification problem. There are many, many more ADPs that do not result in a synapse, however, resulting in unbalanced classes. Because of this, the competition will used the balanced accuracy metric to determine the public and private leaderboard winners. Note that during the competition, submissions will be scored according to the public leaderboard and the private leaderboard will only be revealed at the close of the competition.

This competition is part of a major ongoing research project in neuroscience. Being able to predict when an ADP will convert to a synapse and explain why this happens, is an open problem in neuroscience and a new solution would represent a major advance.

Load In and Merge Data

Concatenate the feature weights and morphological embeddings into np.array's to decrease memory usage, then merge the feature weights and morphological embeddings into the training data.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#load in training data on each potential synapse
data = pd.read_csv("./train_data.csv")

#load in additional features for each neurona
feature_weights = pd.read_csv("./feature_weights.csv")
morph_embeddings = pd.read_csv("./morph_embeddings.csv")
```

In [2]:
```python
# join all feature_weight_i columns into a single np.array column
feature_weights["feature_weights"] = (
    feature_weights.filter(regex="feature_weight_")
    .sort_index(axis=1)
    .apply(lambda x: np.array(x), axis=1)
)
# delete the feature_weight_i columns
feature_weights.drop(
    feature_weights.filter(regex="feature_weight_").columns, axis=1, inplace=True
)

# join all morph_embed_i columns into a single np.array column
morph_embeddings["morph_embeddings"] = (
    morph_embeddings.filter(regex="morph_emb_")
    .sort_index(axis=1)
    .apply(lambda x: np.array(x), axis=1)
)
# delete the morph_embed_i columns
morph_embeddings.drop(
    morph_embeddings.filter(regex="morph_emb_").columns, axis=1, inplace=True
)
```

In [3]:
```python
data = (
    data.merge(
        feature_weights.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        feature_weights.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
)
```

Quantifying the Data

Characteristics of our dataset:

- There are a total of 185,832 potential synapses. Out of these,
  - We have 1,366 that **are** connected (synaptic connections)
  - We have 184,446 that **are not** connected (did not result in a synapse)

- Out of these 1,366 synapses, they result from 77 pre-synaptic neurons and 2,663 post-synaptic neurons

Clearly, the dataset is extremely unbalanced. We will have to employ resampling techniques like undersampling or oversampling to deal with this, and/or focus on ensemble methods that would be effective at dealing with misclassification.

In [4]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from tqdm import tqdm
from sklearn.metrics.pairwise import cosine_similarity

print("Number Connected (Synapses):",data['connected'].sum())

print("\nNumber Not Connected:",(~data['connected']).sum())

print(f"\nAll the adps are from {data['pre_nucleus_id'].nunique()} pre- neurons and {dat
```

```
Number Connected (Synapses): 1366

Number Not Connected: 184466

All the adps are from 77 pre- neurons and 2663 post- neurons.
```

DataFrame

Our dataset comes with 34 features (30 + our 4 new merged features) that we can better understand in the data documentation PDF file.

The features so far include spatial coordinates, the brain area of the neurons (AL, RL, and V1), metrics of neuronal response (oracle - response reliability to repeated presentation; test_score - how well the model predicts response based on input), morphological embeddings, distance metrics, the ID of the neuron, and whether a connection (synapse) is present or not.

In [17]:
```
print("Data Size:",data.shape)
print("\nData:")
data.info()
```

```
Data Size: (185832, 34)

Data:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185832 entries, 0 to 185831
Data columns (total 34 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   ID                              185832 non-null  int64  
 1   axonal_coor_x                   185832 non-null  int64  
 2   axonal_coor_y                   185832 non-null  int64  
 3   axonal_coor_z                   185832 non-null  int64  
 4   dendritic_coor_x                185832 non-null  int64  
 5   dendritic_coor_y                185832 non-null  int64  
 6   dendritic_coor_z                185832 non-null  int64  
 7   adp_dist                        185832 non-null  float64
 8   post_skeletal_distance_to_soma  185832 non-null  float64
 9   pre_skeletal_distance_to_soma   185832 non-null  float64
 10  pre_oracle                      185832 non-null  float64
 11  pre_test_score                  185832 non-null  float64
 12  pre_rf_x                        185832 non-null  float64
 13  pre_rf_y                        185832 non-null  float64
 14  post_oracle                     185832 non-null  float64
 15  post_test_score                 185832 non-null  float64
 16  post_rf_x                       185832 non-null  float64
 17   post_rf_y                        185832 non-null  float64
 18   compartment                      185832 non-null  object
 19   pre_brain_area                   185832 non-null  object
 20   post_brain_area                  185832 non-null  object
 21   pre_nucleus_x                    185832 non-null  int64
 22   pre_nucleus_y                    185832 non-null  int64
 23   pre_nucleus_z                    185832 non-null  int64
 24   post_nucleus_x                   185832 non-null  int64
 25   post_nucleus_y                   185832 non-null  int64
 26   post_nucleus_z                   185832 non-null  int64
 27   pre_nucleus_id                   185832 non-null  int64
 28   post_nucleus_id                  185832 non-null  int64
 29   connected                        185832 non-null  bool
 30   pre_feature_weights              185832 non-null  object
 31   post_feature_weights             185832 non-null  object
 32   pre_morph_embeddings             138123 non-null  object
 33   post_morph_embeddings            185832 non-null  object
dtypes: bool(1), float64(11), int64(15), object(7)
memory usage: 48.4+ MB
```

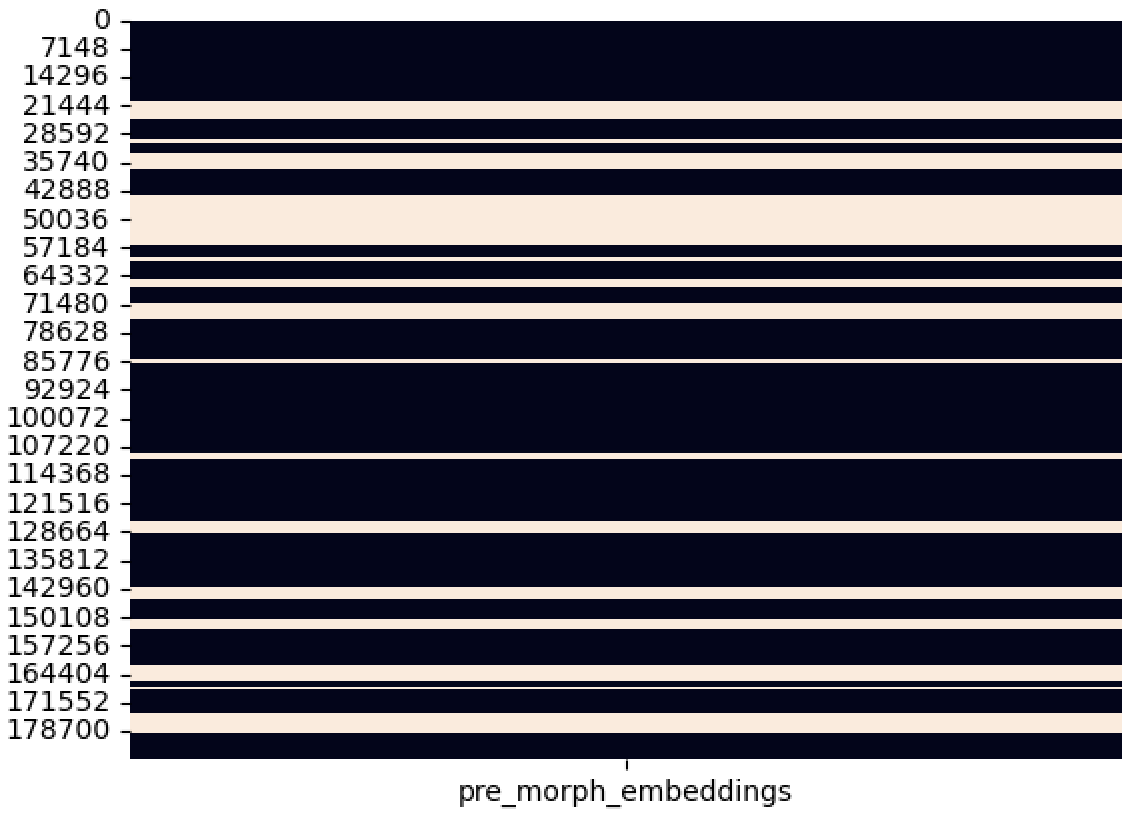
As we see, not all features have the same amount of values.

For instace, pre_morph_embeddings has a significant amount of missing values:

In [117…

```python
# Check for missing values in pre_morph_embeddings
missing_pre_morph = data['pre_morph_embeddings'].isnull()

# Print the number of missing values
print(f"Number of missing values in pre_morph_embeddings: {missing_pre_morph.sum()}")

sns.heatmap(data[['pre_morph_embeddings']].isnull(), cbar=False)
plt.show()
```

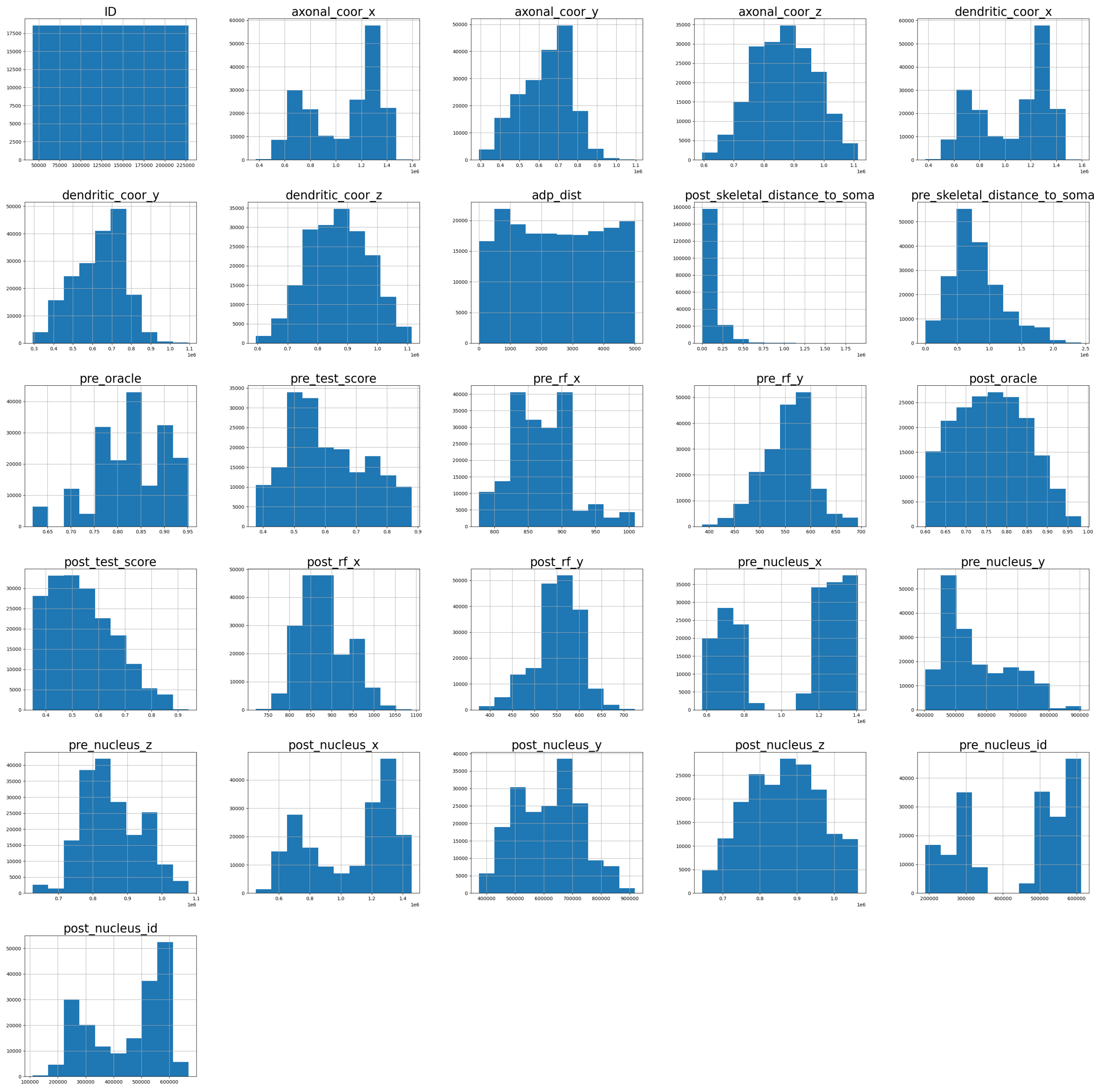
```
Number of missing values in pre_morph_embeddings: 47709
```



In [16]:

```python
# Plot histograms of each column in the DataFrame
axes = data.hist(figsize=(40, 40))

# Loop through each plot and set the title size
```

```
for ax in axes.flatten():
    ax.title.set_size(25)

plt.show()
```



FEATURE ENGINEERING

</br>   In order to capture important aspects of the data that lead to improved model performance, we will have to perform feature engineering while also being aware of model complexity.

One-Hot Encoding

Given our dataset, we can perform one-hot encoding to convert categorical data values into numeric values. We can use these conversions for later when we are engineering new features.

So, we will one-hot encode the following features:

- **brain_area**: It refers to the brain area of which the neuron belongs to
  - We will get:
    - The V1 (Primary Visual Area)
    - The AL (Anterolateral Visual Area)
    - The RL (Rostrolateral Visual Area)

- **compartment**: It refers to the label of neuronal compartment where ADP resides on the post-synaptic neuron
  - We will get:
    - Axon
    - Oblique
    - Apical
    - Basal
    - Soma
    - Apical_tuft
    - Apical_shaft

In [4]:
```
from sklearn.model_selection import train_test_split

# Create a copy of the 'compartment' column
data['compartment_copy'] = data['compartment']

# Perform one-hot encoding on the 'compartment' column
data = pd.get_dummies(data, columns=['compartment'])

# One-hot encoding for 'pre_brain_area' and 'post_brain_area' features
data = pd.get_dummies(data, columns=['pre_brain_area', 'post_brain_area'])

# Split your data into training and test sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=1)
```

Distance Features

We already have some distance metrics:

- **adp_dist**: Distance between axonal ADP coord. and dendritic ADP coord.
- **pre_skeletal_distance_to_soma**: Distance (on the pre-synaptic neuron) from soma to the specific point on its axon that is close to the dendrite of another neuron.
- **post_skeletal_distance_to_soma**: Distance (on the post-synaptic neuron) from soma to the specific point on its dendrite that is close to the dendrite of another neuron.

Note: These features are important since the time it takes for a signal to travel can depend on the distance. If that distance is too large, the signal may degrade, and a synapse could fail to occur.

We will engineer the following distance features:

- **fw_distance**: Euclidean distance between feature weight vectors of pre- and post-synaptic neurons
  - How similar/dissimilar the tuning functions are of the two neurons; smaller dist. -> more similar tuning functions, bigger dist. -> less similar tuning functions

- **axon_dendrite_dist**: Euclidean distance of the physical gap between the axon of the pre-synaptic neuron and the dendrite of the post-synaptic neuron.
  - Note: ADP distances may not precisely match the direct Euclidean distances between axonal and dendritic coordinates, but they exhibit a high degree of correlation.
- **nucleus_dist**: Euclidean distance between nucleus of the pre-synaptic neuron and nucleus of the post-synaptic neuron.
  - Represents how far apart the two neurons are (with respect to their nuclei)
- **axon_dendrite_dist_manhattan**: Manhattan distance of the physical gap between the axon of the pre-synaptic neuron and the dendrite of the post-synaptic neuron.
  - Manhattan distance is better when dealing with sparse data and is more robust to outliers.
- **nucleus_dist_manhattan**: Manhattan distance between nucleus of the pre-synaptic neuron and nucleus of the post-synaptic neuron.
  - Same reason here
- **rf_dist**: Euclidean distance between receptive field locations of the pre- and post-synaptic neurons.
  - The receptive field deals with the specific region in the visual field where a stimulus causes a response
  - Neurons with closer receptive fields may lead to synapses
  - Distance between these fields impacts time it takes a signal to travel between two neurons, which could also impact synaptic activity.
- **pre/post_nucleus_rf_dist**: Euclidean distance between the nucleus and the receptive field locations of the pre- and post-synaptic neurons respectively
  - This distance deals with the pre- and post-synaptic neurons, looking at the distance between the nucleus and the specific area that causes neuronal activation
  - If the receptive field is far, the neuron may be less likely to form a synapse than for closer receptive fields

---

Did not end up using:

- **morph_distance**: Euclidean distance between the morphological embeddings of the pre- and post-synaptic neurons
  - Morphological embeddings deal with the shape/structure of the neurons
  - Two neurons with similar structure or "morphology" may be more likely to form a synapse than less similar structures.

In [5]:
```python
def row_feature_distance(row):
    pre = row["pre_feature_weights"]
    post = row["post_feature_weights"]
    return np.sqrt(((pre - post)**2).sum())

data["fw_distance"] = data.apply(row_feature_distance, axis=1)

# Compute Euclidean distance
data['axon_dendrite_dist'] = np.sqrt(
    (data['axonal_coor_x'] - data['dendritic_coor_x'])**2 +
    (data['axonal_coor_y'] - data['dendritic_coor_y'])**2 +
    (data['axonal_coor_z'] - data['dendritic_coor_z'])**2
)
data['nucleus_dist'] = np.sqrt(
    (data['pre_nucleus_x'] - data['post_nucleus_x'])**2 +
    (data['pre_nucleus_y'] - data['post_nucleus_y'])**2 +
    (data['pre_nucleus_z'] - data['post_nucleus_z'])**2
)

# Compute Manhattan distance
data['axon_dendrite_dist_manhattan'] = (
    abs(data['axonal_coor_x'] - data['dendritic_coor_x']) +
    abs(data['axonal_coor_y'] - data['dendritic_coor_y']) +
    abs(data['axonal_coor_z'] - data['dendritic_coor_z'])
)
data['nucleus_dist_manhattan'] = (
    abs(data['pre_nucleus_x'] - data['post_nucleus_x']) +
    abs(data['pre_nucleus_y'] - data['post_nucleus_y']) +
    abs(data['pre_nucleus_z'] - data['post_nucleus_z'])
)

# Compute Euclidean distance between receptive field locations
data['rf_dist'] = np.sqrt(
    (data['pre_rf_x'] - data['post_rf_x'])**2 +
    (data['pre_rf_y'] - data['post_rf_y'])**2
)

# Compute Euclidean distance between nucleus and receptive field locations
data['pre_nucleus_rf_dist'] = np.sqrt(
    (data['pre_nucleus_x'] - data['pre_rf_x'])**2 +
    (data['pre_nucleus_y'] - data['pre_rf_y'])**2
)
data['post_nucleus_rf_dist'] = np.sqrt(
    (data['post_nucleus_x'] - data['post_rf_x'])**2 +
    (data['post_nucleus_y'] - data['post_rf_y'])**2
)
```

Similarity Features

We will engineer the following similarity features:

- **fw_similarity**: Cosine similarity between the feature weight vectors of the pre- and post-synaptic neurons
  - Cosine similarity quantifies the similarity between two vectors.
  - Recall, the feature weight vectors deal with each neuron's tuning function. If two neurons have similar response patterns (similar feature weights), they might be more likely to form a synapse

- **rf_similarity**: Cosine similarity between receptive field locations of the pre- and post-synaptic neurons
  - This would provide a measure of how similar the receptive field locations are for the pre- and post-synaptic neurons.

In [6]:
```
#cosine similarity function
def row_feature_similarity(row):
    pre = row["pre_feature_weights"]
    post = row["post_feature_weights"]
    return (pre * post).sum() / (np.linalg.norm(pre) * np.linalg.norm(post))

# compute the cosine similarity between the pre- and post- feature weights
data["fw_similarity"] = data.apply(row_feature_similarity, axis=1)

def row_rf_similarity(row):
    pre_rf = np.array([row['pre_rf_x'], row['pre_rf_y']])
    post_rf = np.array([row['post_rf_x'], row['post_rf_y']])
    return (pre_rf * post_rf).sum() / (np.linalg.norm(pre_rf) * np.linalg.norm(post_rf))
```

```
# Compute the cosine similarity between the pre- and post- receptive fields
data['rf_similarity'] = data.apply(row_rf_similarity, axis=1)
```

Interaction Features

We will engineer the following interaction features:

- **pre/post_oracle_adp_dist**: Captures how the reliability of neural activation (of the pre- and post-synaptic neuron) changes with the distance between the axon and dendrite
  - Recall, the oracle feature deals with neuronal response reliability to repeated visual stimulus
  - So, neuronal response reliability could vary for different distances for both pre- and post-synaptic neurons

- **pre/post_oracle_pre_rf_x/y_interaction**: Captures how the reliability of neural activation (of the post-synaptic neuron) changes with the receptive field location, measured by rf_x & rf_y
  - So, neuronal response reliability could vary for different receptive fields for both pre- and post-synaptic neurons

- **pre/post_oracle_axon_dendrite_dist_interaction**: Captures how the reliability of neural activation (of the post-synaptic neuron) changes with the Euclidean distance between the pre-synaptic neuron's axon and the post-synaptic neuron's dendrite.
  - So, neuronal response reliability could vary for different Eucliean distances for both pre- and post-synaptic neurons
  - Remember, ADP distances may not precisely match the direct Euclidean distances between axonal and dendritic coordinates, but they exhibit a high degree of correlation

- **pre/post_oracle_test_score_interaction**: Represents the combined effect of a neuron's response reliability and the predictive performance of the deep learning model on withheld test trials
  - Recall, the test_score is a measure of how well the deep learning model could predict the outcomes
  - So, this feature is important since a neuron's ability to form a synapse could be influenced both by response reliability and the performance of the model
  - Neurons with both high activation reliability and high predictive performance could form a synapse when compared to lower values

- **pre_AL/RL/V1_post_RL/V1/AL**: Represents interactions between the specific brain areas (AL, RL, and V1) of the pre-synaptic and post-synaptic neurons
  - Interactions between different brain areas may influence whether we will get a synapse from an ADP

In [7]:
```
# Compute interaction between oracle and distance features
data['pre_oracle_adp_dist'] = data['pre_oracle'] * data['adp_dist']
data['post_oracle_adp_dist'] = data['post_oracle'] * data['adp_dist']

# Compute interaction between oracle and receptive field information
data['pre_oracle_pre_rf_x_interaction'] = data['pre_oracle'] * data['pre_rf_x']
data['pre_oracle_pre_rf_y_interaction'] = data['pre_oracle'] * data['pre_rf_y']
data['post_oracle_post_rf_x_interaction'] = data['post_oracle'] * data['post_rf_x']
data['post_oracle_post_rf_y_interaction'] = data['post_oracle'] * data['post_rf_y']

# Compute interaction between pre_oracle and axon_dendrite_dist
data['pre_oracle_axon_dendrite_dist_interaction'] = data['pre_oracle'] * data['axon_dend
data['post_oracle_axon_dendrite_dist_interaction'] = data['post_oracle'] * data['axon_de

# Compute interaction between oracle and test score
```

```
data['pre_oracle_test_score_interaction'] = data['pre_oracle'] * data['pre_test_score']
data['post_oracle_test_score_interaction'] = data['post_oracle'] * data['post_test_score

# Create interaction features
data['pre_AL_post_RL'] = data['pre_brain_area_AL'] * data['post_brain_area_RL']
data['pre_AL_post_V1'] = data['pre_brain_area_AL'] * data['post_brain_area_V1']
data['pre_RL_post_AL'] = data['pre_brain_area_RL'] * data['post_brain_area_AL']
data['pre_RL_post_V1'] = data['pre_brain_area_RL'] * data['post_brain_area_V1']
data['pre_V1_post_AL'] = data['pre_brain_area_V1'] * data['post_brain_area_AL']
data['pre_V1_post_RL'] = data['pre_brain_area_V1'] * data['post_brain_area_RL']
```

Difference Features

We will engineer only one difference feature:

- **test_score_diff**: Represents the difference in predictive performance of the deep learning predictive model between the pre- and post-synaptic neuron
  - If the difference in test scores is significant, it might influence whether or not a synapse will form
  - A higher score may lead to a higher likelihood of a synapse

In [8]:
```
data['test_score_diff'] = data['post_test_score'] - data['pre_test_score']
```

Number Count Features

We will engineer the following count features:

- **total_synapses**: Represents the total number of synapses formed (from each pre-synaptic neuron)
  - The number of synapses a pre-synaptic neuron (the one that sends the signal) forms could potentially influnce its overall connectivity.
  - In other words, a neuron with a larger number of synapses could have a higher possibility in general of forming a synapse than those with lower number of synapses.

- **pre/post_(compartment)_count**: The count of each type of compartment (axon, oblique, apical, etc.) for the pre- and post-synaptic neurons
  - Recall, compartment refers to the label of neuronal compartment where ADP resides on the post-synaptic neuron.
  - Taking into account the specific number of compartments per neuron could determine its likelihood in being part of a synapse

- **compartment_synapse_prop**: The proportion of synapses in each neuronal compartment; quantifies the likelihood of synapse formation in each compartment.
  - Certain types of compartments may be likely to lead to a synapse than others. The propensity for synapse formation could vary between the compartments

In [9]:
```
# Calculate the total number of synapses for each neuron
total_synapses = data.groupby('pre_nucleus_id').size()

# Add the Total Synapses to your DataFrame
data['total_synapses'] = data['pre_nucleus_id'].map(total_synapses)

# List of all compartment types
compartments = ['apical', 'apical_shaft', 'apical_tuft', 'axon', 'basal', 'oblique', 'so

# Create new aggregated features for pre_nucleus_id
for compartment in compartments:
    # Create new feature name
    new_feature = f'pre_{compartment}_count'

    # Create new feature
    data[new_feature] = data.groupby('pre_nucleus_id')[f'compartment_{compartment}'].tra

# Create new aggregated features for post_nucleus_id
for compartment in compartments:
    # Create new feature name
    new_feature = f'post_{compartment}_count'

    # Create new feature
    data[new_feature] = data.groupby('post_nucleus_id')[f'compartment_{compartment}'].tr

# Calculate the proportion of synapses in each compartment
synapse_proportions = data.groupby('compartment_copy')['connected'].mean()

# Map the proportions to the original DataFrame
data['compartment_synapse_prop'] = data['compartment_copy'].map(synapse_proportions)
```

Engineered Features That Were Not Used

There are a number of engineered features that ended up either not making a difference in model performance, made model performance worse, or were misleading in the sense that they scored incredibly well in Jupyter Notebook but performed horribly in Kaggle.

Notes:

- One group of features that were incredibly difficult to work with was the morphological embeddings. Using techniques like imputation to deal with the missing values of pre_morph_embeddings was harder than expected, and even when new features were engineered from the morphological embeddings (morph_distance and morph_similarity), they did not perform well on the model.

- The interaction features below failed to make a meaningful difference in the model, and any change in performance was usually negative. They also made the model more complex for no specific benefit.

Each commented out block of code in triple quotes represents a failed feature

In [28]:
```
### DISTANCE FEATURES ###

"""
# DID NOT USE
# Euclidean distance function for morphological embeddings
#def row_morph_distance(row):
#    pre = row["pre_morph_embeddings"]
#    post = row["post_morph_embeddings"]
#    return np.linalg.norm(pre - post)

# Compute the Euclidean distance between the pre- and post- morphological embeddings
#data["morph_distance"] = data.apply(row_morph_distance, axis=1)
"""

### SIMILARITY FEATURES ###

"""
# DID NOT USE
# Cosine similarity function for morphological embeddings
#def row_morph_similarity(row):
#    pre = row["pre_morph_embeddings"]
#    post = row["post_morph_embeddings"]
#    return (pre * post).sum() / (np.linalg.norm(pre) * np.linalg.norm(post))

# Compute the cosine similarity between the pre- and post- morphological embeddings
#data["morph_similarity"] = data.apply(row_morph_similarity, axis=1)
"""

### INTERACTION FEATURES ###

"""
# Compute interaction between post_oracle and nucleus_dist
# data['pre_oracle_nucleus_dist_interaction'] = data['pre_oracle'] * data['nucleus_dist'
# data['post_oracle_nucleus_dist_interaction'] = data['post_oracle'] * data['nucleus_dis
"""

"""
# Compute interaction between adp_dist and other features
# data['adp_dist_axon_dendrite_dist_interaction'] = data['adp_dist'] * data['axon_dendri
# data['adp_dist_nucleus_dist_interaction'] = data['adp_dist'] * data['nucleus_dist'] ->
# data['adp_dist_rf_dist_interaction'] = data['adp_dist'] * data['rf_dist'] -> DID NOT U
"""

"""
# Create a new feature that is True if both pre_oracle and post_oracle are at least 0.8
# data['high_reliability'] = (data['pre_oracle'] >= 0.5) & (data['post_oracle'] >= 0.5)
"""

### DIFFERENCE FEATURES ###

"""
#data['test_score_ratio'] = data['post_test_score'] / data['pre_test_score'] -> DID NOT
#data['test_score_avg'] = (data['post_test_score'] + data['pre_test_score']) / 2 -> DID
"""

"""
# Compute difference between pre and post features
# data['oracle_diff'] = data['post_oracle'] - data['pre_oracle']
# data['rf_x_diff'] = data['post_rf_x'] - data['pre_rf_x']
# data['rf_y_diff'] = data['post_rf_y'] - data['pre_rf_y']
"""

"""
# data['oracle_int'] = data['post_oracle'] * data['pre_oracle']
"""

### RATIO FEATURES ###


"""
# List of all compartment types
compartments = ['compartment_apical', 'compartment_apical_shaft', 'compartment_apical_tu

# Create a new feature that represents the total count of compartments for each neuron
data['total_compartments'] = data[compartments].sum(axis=1)

# Create new features that represent the ratio of each compartment type to the total num
for compartment in compartments:
    data[f'{compartment}_ratio'] = data[compartment] / data['total_compartments']
"""

"""
# Synapse ratio feature

# Calculate the total number of synapses for each neuron
total_synapses = data.groupby('pre_nucleus_id').size()

# Calculate the number of connected synapses for each neuron
```

```
connected_synapses = data[data['connected'] == True].groupby('pre_nucleus_id').size()

# Calculate the Synapse Ratio
synapse_ratio = connected_synapses / total_synapses

# Add the Synapse Ratio to your DataFrame
data['synapse_ratio'] = data['pre_nucleus_id'].map(synapse_ratio)
"""

### COUNT FEATURES ###

"""
# Compute the global mean
mean = data['connected'].mean()

# Compute the number of values and the mean of each group
agg = data.groupby('compartment_copy')['connected'].agg(['count', 'mean'])
counts = agg['count']
means = agg['mean']

# Compute the smoothed means
smooth = (counts * means + 100 * mean) / (counts + 100)

# Replace each compartment with the smoothed mean
# data['compartment_target_enc'] = data['compartment_copy'].map(smooth) -> DID NOT USE

# data['compartment_count'] = data['compartment_copy'].map(data['compartment_copy'].valu
"""

### IMPUTATION ###

# from sklearn.impute import SimpleImputer

# Mean imputation
# mean_imputer = SimpleImputer(strategy='mean')
# data['morph_distance'] = mean_imputer.fit_transform(data['morph_distance'].values.resh
# data['morph_similarity'] = mean_imputer.fit_transform(data['morph_similarity'].values.

# Median imputation
# median_imputer = SimpleImputer(strategy='median')
# data['morph_distance'] = median_imputer.fit_transform(data['morph_distance'].values.re
# data['morph_similarity'] = median_imputer.fit_transform(data['morph_similarity'].value

# Most frequent imputation
# most_frequent_imputer = SimpleImputer(strategy='most_frequent')
# data['morph_distance'] = most_frequent_imputer.fit_transform(data['morph_distance'].va
# data['morph_similarity'] = most_frequent_imputer.fit_transform(data['morph_similarity'

# data['synapse_ratio'] = mean_imputer.fit_transform(data['synapse_ratio'].values.reshap

### OTHER WAYS OF DEALING WITH EMPTY VALUES ###

# Fill NaNs with the mean of the column
# mean_morph_similarity = train_data['morph_distance'].mean()
# train_data['morph_distance'].fillna(mean_morph_similarity, inplace=True)

# Fill NaNs with the mean of the column in the training data
# mean_morph_similarity = test_data['morph_distance'].mean()
# test_data['morph_distance'].fillna(mean_morph_similarity, inplace=True)

# Drop rows with NaNs in the 'morph_similarity' column
# train_data.dropna(subset=['morph_similarity'], inplace=True)
# test_data.dropna(subset=['morph_similarity'], inplace=True)

# Drop rows with NaNs in the 'morph_distance' column
# train_data.dropna(subset=['morph_distance'], inplace=True)
# test_data.dropna(subset=['morph_distance'], inplace=True)

# print(X_resampled.isnull().sum())  # Check for NaNs
# print(np.isinf(X_resampled).sum())  # Check for infinity values
# print((X_resampled > np.finfo(np.float64).max).sum())  # Check for values too large fo
```

Check the Dataframe

We now have 88 features in total.

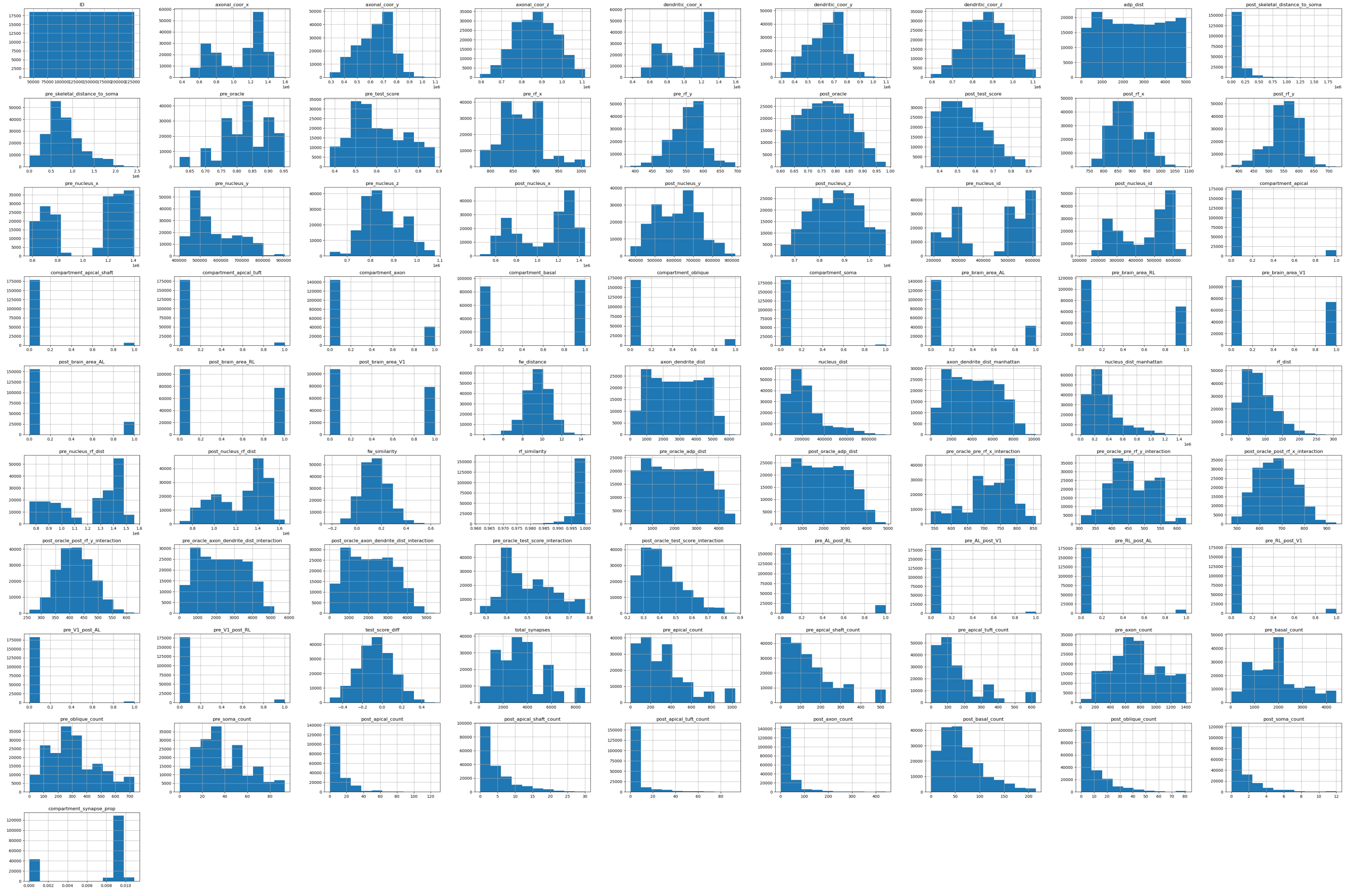
Plotting a histogram for each column in the Dataframe, we can get a better look at each feature's distribution of values

In [13]:
```
print("Main Dataframe Size:",data.shape)
print("\nMain Dataframe:")
data.info()
column_names = data.columns.tolist()
```

```
Main Dataframe Size: (185832, 88)

Main Dataframe:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185832 entries, 0 to 185831
Data columns (total 88 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   ID                                  185832 non-null  int64
 1   axonal_coor_x                       185832 non-null  int64
 2   axonal_coor_y                       185832 non-null  int64
 3   axonal_coor_z                       185832 non-null  int64
 4   dendritic_coor_x                    185832 non-null  int64
 5   dendritic_coor_y                    185832 non-null  int64
 6   dendritic_coor_z                    185832 non-null  int64
 7   adp_dist                            185832 non-null  float64
 8   post_skeletal_distance_to_soma      185832 non-null  float64
 9   pre_skeletal_distance_to_soma       185832 non-null  float64
 10  pre_oracle                          185832 non-null  float64
 11  pre_test_score                      185832 non-null  float64
 12  pre_rf_x                            185832 non-null  float64
 13  pre_rf_y                            185832 non-null  float64
 14  post_oracle                         185832 non-null  float64
 15  post_test_score                     185832 non-null  float64
 16  post_rf_x                           185832 non-null  float64
 17  post_rf_y                           185832 non-null  float64
 18  pre_nucleus_x                       185832 non-null  int64
 19  pre_nucleus_y                       185832 non-null  int64
 20  pre_nucleus_z                       185832 non-null  int64
 21  post_nucleus_x                      185832 non-null  int64
 22  post_nucleus_y                      185832 non-null  int64
 23  post_nucleus_z                      185832 non-null  int64
 24  pre_nucleus_id                      185832 non-null  int64
 25  post_nucleus_id                     185832 non-null  int64
 26  connected                           185832 non-null  bool
 27  pre_feature_weights                 185832 non-null  object
 28  post_feature_weights                185832 non-null  object
 29  pre_morph_embeddings                138123 non-null  object
 30  post_morph_embeddings               185832 non-null  object
 31  compartment_copy                    185832 non-null  object
 32  compartment_apical                  185832 non-null  uint8
 33  compartment_apical_shaft            185832 non-null  uint8
 34  compartment_apical_tuft             185832 non-null  uint8
 35  compartment_axon                    185832 non-null  uint8
 36  compartment_basal                           185832 non-null  uint8
 37  compartment_oblique                         185832 non-null  uint8
 38  compartment_soma                            185832 non-null  uint8
 39  pre_brain_area_AL                           185832 non-null  uint8
 40  pre_brain_area_RL                           185832 non-null  uint8
 41  pre_brain_area_V1                           185832 non-null  uint8
 42  post_brain_area_AL                          185832 non-null  uint8
 43  post_brain_area_RL                          185832 non-null  uint8
 44  post_brain_area_V1                          185832 non-null  uint8
 45  fw_distance                                 185832 non-null  float64
 46  axon_dendrite_dist                          185832 non-null  float64
 47  nucleus_dist                                185832 non-null  float64
 48  axon_dendrite_dist_manhattan                185832 non-null  int64
 49  nucleus_dist_manhattan                      185832 non-null  int64
 50  rf_dist                                     185832 non-null  float64
 51  pre_nucleus_rf_dist                         185832 non-null  float64
 52  post_nucleus_rf_dist                        185832 non-null  float64
 53  fw_similarity                               185832 non-null  float64
 54  rf_similarity                               185832 non-null  float64
 55  pre_oracle_adp_dist                         185832 non-null  float64
 56  post_oracle_adp_dist                        185832 non-null  float64
 57  pre_oracle_pre_rf_x_interaction             185832 non-null  float64
 58  pre_oracle_pre_rf_y_interaction             185832 non-null  float64
 59  post_oracle_post_rf_x_interaction           185832 non-null  float64
 60  post_oracle_post_rf_y_interaction           185832 non-null  float64
 61  pre_oracle_axon_dendrite_dist_interaction   185832 non-null  float64
 62  post_oracle_axon_dendrite_dist_interaction  185832 non-null  float64
 63  pre_oracle_test_score_interaction           185832 non-null  float64
 64  post_oracle_test_score_interaction          185832 non-null  float64
 65  pre_AL_post_RL                              185832 non-null  uint8
 66  pre_AL_post_V1                              185832 non-null  uint8
 67  pre_RL_post_AL                              185832 non-null  uint8
 68  pre_RL_post_V1                              185832 non-null  uint8
 69  pre_V1_post_AL                              185832 non-null  uint8
 70  pre_V1_post_RL                              185832 non-null  uint8
 71  test_score_diff                             185832 non-null  float64
 72  total_synapses                              185832 non-null  int64
 73  pre_apical_count                            185832 non-null  float64
 74  pre_apical_shaft_count                      185832 non-null  float64
 75  pre_apical_tuft_count                       185832 non-null  float64
 76  pre_axon_count                              185832 non-null  float64
 77  pre_basal_count                             185832 non-null  float64
 78  pre_oblique_count                           185832 non-null  float64
 79  pre_soma_count                              185832 non-null  uint8
 80  post_apical_count                           185832 non-null  uint8
 81  post_apical_shaft_count                     185832 non-null  uint8
 82  post_apical_tuft_count                      185832 non-null  uint8
 83  post_axon_count                             185832 non-null  float64
 84  post_basal_count                            185832 non-null  uint8
 85  post_oblique_count                          185832 non-null  uint8
 86  post_soma_count                             185832 non-null  uint8
 87  compartment_synapse_prop                    185832 non-null  float64
dtypes: bool(1), float64(38), int64(18), object(5), uint8(26)
memory usage: 92.7+ MB
```

In [120…

```python
# Plot histograms of each column in the DataFrame
data.hist(figsize=(60, 40))
plt.show()
```



MODELING

</br>   Now, we will work on building a model that optimizes performance but does not lead to overfitting. There are a number of models that will be tested, as well as hyperparameters that will require a lot of tuning.

To start, we will import the simple Logistic Regression classification model from the example notebook to use as reference:

Simple Logistic Regression

Accuracy of approximately 99.27%, but the confusion matrix reveals that it's not performing well on predicting when a synapse will form (True Positives).

It's predicting all ADPs as not forming a synapse. This is likely due to the imbalance in your data, where the majority of ADPs do not form a synapse.

```
In [10]: # logistic regression model (connected ~ fw_similarity + adp)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import balanced_accuracy_score, accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# split into a train and test set
#(Even though we're working with the competition training set, you may want to have your
train_data, test_data = train_test_split(data, test_size=0.2, random_state=1)

# create pipeline
```

```
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", LogisticRegression(random_state=2))]
)

# fit model
pipe.fit(train_data[["fw_similarity", "adp_dist"]], train_data["connected"])

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist"]])[:, 1]

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")
```

```
accuracy: 0.9927085855732236
```

- **True Negatives (Top-Left, 36896)**: These are cases in which model correctly predicted that a synapse would not form.
- **False Positives (Top-Right, 0)**: These are cases in which model incorrectly predicted that a synapse would form.
- **False Negatives (Bottom-Left, 271)**: These are cases in which model incorrectly predicted that a synapse would not form.
- **True Positives (Bottom-Right, 0)**: These are cases in which model correctly predicted that a synapse would form.

In [13]:
```
confusion_matrix(test_data['connected'], test_data['pred'] > .5)
```

Out[13]:
```
array([[36896,     0],
       [  271,     0]], dtype=int64)
```

The Balanced Accuracy is 0.5. This means that the model is performing no better than random guessing.

In [14]:
```
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
balanced accuracy: 0.5
```

Addressing the unbalanced nature of our dataset

- Oversampling to address the imbalance in the data. It involves randomly duplicating examples in the minority class (in this case, ADPs where a synapse forms) to increase their prevalence in the training data.

- Accuracy of model is approximately 64.78%. This is lower but remember that accuracy can be misleading when dealing with imbalanced datasets.

- Balanced Accuracy of your model is approximately 72.19%. The model is doing a better job at classifying both majority and minority classes

In [11]:
```
from imblearn.over_sampling import RandomOverSampler

# oversample connected neuron pairs
ros = RandomOverSampler(random_state=0)
X_resampled, y_resampled = ros.fit_resample(
    train_data[["fw_similarity", "adp_dist"]], train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)
```

```
# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist"]])[:, 1]

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.647752038098313
[[23859 13037]
 [   55   216]]
balanced accuracy: 0.7218517172433216
```

Modeling Notes

We will be using:

- **Resampling**: We will be using undersampling as opposed to oversampling. This involves randomly removing examples from the majoriy class to decrease instances of over-representation. Additionally, it will be incredibly beneficial in terms of computational efficiency/cost.
  - Note, oversampling (SMOTE) *was* used at certain points, but it failed to perform better than undersampling, and took way too long to run.

- **Standardization**: We transform the features of the dataset to have a mean of zero and a standard deviation of one. To do this, we will use StandardScaler()
  - Alternatives like MinMaxScaler and MaxAbsScaler did not yield to better results.
  - StandardScaler is also less sensitive to extreme outliers

- **Pipeline**: Allows a linear series of data transforms to be linked together. Streamlines the machine learning workflow
  - They simplify the process of fitting and transforming data, and can help prevent data leakage by ensuring preprocessing steps happen only on the training data
  - The workflow also becomes more easily reproducible

- **Confusion Matrix**: Represents the accuracy of a classification model. Displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN)
  - We can use a confusion matrix in every output to understand how many predictions were correct and the kind of errors our model made
  - Can better understand where the model became 'confused'

Support Vector Machines (SVMs)

We will start with a simple SVM with default parameters (meaning our kernel will be RBF), and train this SVM only on two simple features: `fw_similarity` and `adp_dist`. Doing so, we will see worse results than the logistic regression from earlier on model accuracy, however a slightly better result in terms of balanced accuracy.

In [16]:
```
from sklearn.svm import SVC
from sklearn.metrics import balanced_accuracy_score, accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler
```

In [17]:
```
# split into a train and test set
train_data, test_data = train_test_split(data, test_size=0.2, random_state=1)

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(probability=True, random_state=2))]
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist"]], train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist"]])[:, 1]

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6129900180267441
[[22555 14341]
 [   43   228]]
balanced accuracy: 0.7263206463645295
```

However,

Simply going from `rbf` to a `linear` kernel, we can see a massive improvement in model accuracy, but a worse balanced accuracy.

Note: After manually trying all possible kernels (linear, poly, rbf, sigmoid), linear massively outperformed all other kernels

In [13]:
```
# split into a train and test set
train_data, test_data = train_test_split(data, test_size=0.2, random_state=1)

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', probability=True, rand
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist"]], train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist"]])[:, 1]

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6563887319396239
[[24184 12712]
 [   59   212]]
balanced accuracy: 0.7188759149083251
```

At this point of the competition, there were two new features that were engineered: `axon_dendrite_dist` and `nucleus_dist`

Adding these two to our pipeline and training them on our linear SVM, we see performance on both metrics improve drastically.

Our Kaggle public leaderboard score was around **~70.12%**

In [14]:
```
# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', probability=True, rand
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist"]],
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))
```

```
# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6817068905211613
[[25121 11775]
 [   55   216]]
balanced accuracy: 0.7389538421349088
```

Introducing three new engineered features led to increased performance on both metrics. Those three being `rf_dist`, `pre_oracle_adp_dist`, and `post_oracle_adp_dist`

In [17]:

```
# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', probability=True, rand
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6831328867005677
[[25170 11726]
 [   51   220]]
balanced accuracy: 0.7469979445566355
```

Hyperparameter Tuning

We will use both GridSearchCV and RandomizedSearchCV to find the optimal parameters

Note: With RandomizedSearchCV, there will be a tradeoff between runtime and best possible solution in the n_iter parameter

In [45]:

```
from sklearn.model_selection import GridSearchCV

# define the parameter grid
param_grid = {
    'model__C': [0.01, 0.1, 1, 10],
    'model__gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1, 10, 100],
}
```

```
# create a GridSearchCV object
grid = GridSearchCV(pipe, param_grid, cv=5, scoring='balanced_accuracy')

# fit the model and tune parameters
grid.fit(X_resampled, y_resampled)

print(f"Best parameters: {grid.best_params_}")
print(f"Best balanced accuracy: {grid.best_score_}")
```

```
Best parameters: {'model__C': 10, 'model__gamma': 'scale'}
Best balanced accuracy: 0.7351598173515981
```

In [47]:
```
from scipy.stats import uniform, randint

# define the parameter distribution
param_dist = {
    'model__C': uniform(0.001, 50),
    'model__gamma': uniform(0.001, 10),
}

# create a RandomizedSearchCV object
random_search = RandomizedSearchCV(pipe, param_distributions=param_dist,
                                   n_iter=10, cv=5, scoring='balanced_accuracy', random_

# fit the model and tune parameters
random_search.fit(X_resampled, y_resampled)

print(f"Best parameters: {random_search.best_params_}")
print(f"Best balanced accuracy: {random_search.best_score_}")
```

```
Best parameters: {'model__C': 27.484123893935458, 'model__gamma': 4.354223926182769}
Best balanced accuracy: 0.7356164383561644
```

Manipulating one hyperparameter, *C*, yielded a significant difference in performance.

Specifically, making C=10 increased both model accuracy and balanced accuracy by around ~0.002

Notes:

- *Increasing C to 10 (from a default of 1.0) means we are increasing the penalty for misclassified points (making our margin narrower). Our SVM will try to classify **all** points correctly, including the outliers. We can see the number of true positives increase and false positives decrease in our confusion matrix.*
- *RandomizedSearch gave us C to be 27, but to avoid any potential overfitting, C was chosen to be 10. Even if RandomizedSearchCV had C to be bigger than 10, there were not really any major advances in accuracy*
- *Gamma was best at **scale**, which is the default parameter setting.*

At this point, the highest Kaggle score was around **~70.55%**

In [35]:
```
# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', C=10.0, probability=Tr
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)
```

```
# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6853929561169855
[[25254 11642]
 [   51   220]]
balanced accuracy: 0.7481362793354733
```

Adding one-hot encoded brain area features

After one-hot encoding the brain area categorical features, we engineered some features looking at their interactions. Those features were `pre_AL/RL/V1_post_RL/V1/AL`

We can see a strong increase in model performance, but a dip in balanced accuracy.

Although there was a dip in the balanced accuracy, we still managed to end up with a Kaggle score of **~70.55%**

In [56]:
```
# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', C=10.0, probability=Tr
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6879220814163102
[[25354 11542]
```

```
 [   57   214]]
balanced accuracy: 0.7384213290853636
```

Our Best Linear SVM

We get these results from the various new engineered features

---

Results:

We get a model accuracy of **~69.155%** and a balanced accuracy of **~73.476%**

Kaggle *Public* Score -> **70.877%**

In [164…

```
# Split your data into training and test sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=1)

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', C=10.0, probability=Tr
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.691554335835553
[[25492 11404]
 [   60   211]]
balanced accuracy: 0.7347563951571866
```

In [ ]:

```
from imblearn.combine import SMOTEENN
from sklearn.svm import SVC


# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", SVC(kernel='linear', C=10.0, probability=Tr
)

# apply SMOTEENN
smote_enn = SMOTEENN(random_state=0)
X_resampled, y_resampled = smote_enn.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['p
```

Random Forests

As the number of engineered features would continue to increase, there slowly became a worry of overfitting due to increased model complexity. Thus, there was a transition from SVMs to ensemble methods, as they tend to be robust to overfitting.

So, we start with a simple Random Forest algorithm:

- The only parameter changed so far was **class_weight**, and it was changed to *balanced* because of our imbalanced dataset.
- We also used a small subset of features to start.
- Performance on both metrics (model accuracy and balanced accuracy) plummeted.

In [27]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, confusion_matrix, balanced_accuracy_score
import time
```

In [71]:

```
# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", RandomForestClassifier(class_weight='balanc
)

# undersample majority class using RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
```

```
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6621465278338311
[[24404 12492]
 [   65   206]]
balanced accuracy: 0.7107871571994124
```

Note: Oversampling failed miserably. It gave incredible model accuracy but awful balanced accuracy, making the following models essentially useless:

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import ADASYN

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()), ("model", RandomForestClassifier(class_weight='balanc
)

# oversample minority class using ADASYN
adasyn = ADASYN(random_state=0)
X_resampled, y_resampled = adasyn.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.9380364301665456
[[34808  2088]
 [  215    56]]
balanced accuracy: 0.5750252829935064
```

In [62]:
```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

# define pipeline
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("over", SMOTE(sampling_strategy=0.5)),  # over-sample the minority class to 50%
    ("under", RandomUnderSampler(sampling_strategy=0.5)),  # under-sample the majority c
    ("model", RandomForestClassifier(random_state=1, n_estimators=100, bootstrap=False))
])

# fit model
pipe.fit(train_data[["fw_similarity", "fw_distance", "compartment_synapse_prop", "axon_d
train_data["connected"])

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "fw_distance", "compa

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.9912825893938171
[[36837    59]
 [  265     6]]
balanced accuracy: 0.5102705660350185
```

The Second Pick

As time would pass, many new features were engineered. Those features were tested on this Random Forest.

In the following RF model, all aforementioned features, except for those we did not use (morph_similarity, morph_distance, etc.) and except for any features relating to compartments, were used to train and test this model.

This model ended up becoming my second pick in Kaggle

Out of fear that my main model would perform poorly on new data, and may have overfit to the public leadboard data, this RF model was used as a backup since Random Forests are robust to overfitting.

Results:

We get a model accuracy of **~73.062%** and a balanced accuracy of **~77.641%**

Kaggle *Public* Score -> **74.295%**\ Kaggle *Private* Score -> **77.900%**

Note: GridSearchCV was not helpful. The "optimal hyperparameters" it gave me performed worse than simply leaving n_estimators to 100 and bootstrap to False. However, cv was put to 3 for computational efficiency, so it's possible a cv of 5 or 10 would've yielded different results.

In [38]:
```
# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()),
     ("model", RandomForestClassifier(random_state=1, n_estimators=100, bootstrap=False)
)

start_time = time.time()

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "fw_distance", "compartment_synapse_prop", "axon_dendri
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "fw_distance", "compa

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)

# Stop timer and print time
end_time = time.time()
print(f"Time taken: {end_time - start_time} seconds")
```

```
accuracy: 0.730621250033632
[[26932  9964]
 [   48   223]]
balanced accuracy: 0.7764109270537631
Time taken: 8.174513578414917 seconds
```

In [74]:
```
from sklearn.model_selection import GridSearchCV

# define the parameter grid
param_grid = {
    'model__n_estimators': [10, 50, 100],
    'model__max_depth': [10, 20, 30, 40, 50, None],
    'model__min_samples_split': [2, 5, 10],
    'model__min_samples_leaf': [1, 2, 4],
    'model__bootstrap': [True, False]
}

# create a GridSearchCV object
grid_search = GridSearchCV(estimator=pipe, param_grid=param_grid, cv=3, verbose=2, n_job

# fit the model
grid_search.fit(X_resampled, y_resampled)

# print the best parameters
print(grid_search.best_params_)

# predict on test data using the best model
test_data["pred"] = grid_search.predict_proba(test_data[["fw_similarity", "fw_distance",

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")
```

```
# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
Fitting 3 folds for each of 324 candidates, totalling 972 fits
{'model__bootstrap': True, 'model__max_depth': 10, 'model__min_samples_leaf': 2, 'model_
_min_samples_split': 10, 'model__n_estimators': 100}
accuracy: 0.6970699814351441
[[25674 11222]
 [   37   234]]
balanced accuracy: 0.7796582115322455
```

In [39]:
```
# Get feature importances
importances = pipe.named_steps['model'].feature_importances_

# Convert importances into a pandas Series for easier handling
importances_series = pd.Series(importances, index=X_resampled.columns.tolist())

# Sort importances
sorted_importances = importances_series.sort_values(ascending=False)

# Print the sorted importances
print(sorted_importances)
```

```
adp_dist                                         0.079954
compartment_synapse_prop                         0.067981
pre_oracle_axon_dendrite_dist_interaction        0.063369
pre_oracle_adp_dist                              0.062627
post_oracle_adp_dist                             0.061688
axon_dendrite_dist_manhattan                     0.059267
post_skeletal_distance_to_soma                   0.052325
axon_dendrite_dist                               0.050342
post_oracle_axon_dendrite_dist_interaction       0.042476
nucleus_dist                                     0.040098
nucleus_dist_manhattan                           0.037912
pre_skeletal_distance_to_soma                    0.035993
fw_similarity                                    0.033663
post_nucleus_rf_dist                             0.033283
rf_dist                                          0.030249
post_oracle_post_rf_x_interaction                0.028154
fw_distance                                      0.027800
post_oracle_post_rf_y_interaction                0.027233
test_score_diff                                  0.025689
post_oracle_test_score_interaction               0.024630
total_synapses                                   0.022520
pre_nucleus_rf_dist                              0.020284
pre_oracle_pre_rf_x_interaction                  0.018277
pre_oracle_test_score_interaction                0.017937
pre_oracle_pre_rf_y_interaction                  0.015382
post_brain_area_V1                               0.004606
pre_brain_area_RL                                0.002374
post_brain_area_RL                               0.002371
pre_brain_area_V1                                0.002123
pre_AL_post_RL                                   0.001825
post_brain_area_AL                               0.001752
pre_brain_area_AL                                0.001578
pre_RL_post_AL                                   0.001376
pre_RL_post_V1                                   0.001273
pre_V1_post_RL                                   0.000589
pre_AL_post_V1                                   0.000512
pre_V1_post_AL                                   0.000490
dtype: float64
```

Model Stacking

Model stacking was used thanks to its popularity in machine learning. We are essentially improving predictive performance by combining the predictions of multiple models. Using diverse models can also help the stacked model be more robust and may end up generalizing better to new data.

There also was an attempt to use **PCA** for dimensionality reduction. However, as more work was done on the competition, PCA proved to not be as useful as originally thought.

To start:

- Random classifiers were used with default parameters to see what we would get.
- Out final estimator was Logistic Regression

Results:

We get a model accuracy of **~66.903%** and a balanced accuracy of **~73.440%**

In [92]:
```
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import ADASYN
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import StackingClassifier, AdaBoostClassifier, GradientBoostingCla
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from imblearn.over_sampling import ADASYN
from xgboost import XGBClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
```

In [21]:
```
from sklearn.ensemble import StackingClassifier, AdaBoostClassifier, GradientBoostingCla
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from imblearn.over_sampling import ADASYN

# define base models
base_models = [
    ("svc", SVC(kernel='linear', C=10.0, probability=True, random_state=2)),
    ("ada", AdaBoostClassifier(random_state=2)),
    ("gbc", GradientBoostingClassifier(random_state=2)),
    ("lr", LogisticRegression()),
    ("knn", KNeighborsClassifier()),
    ("rf", RandomForestClassifier(random_state=1))
]

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()),
     ("model", StackingClassifier(estimators=base_models, final_estimator=LogisticRegres
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.6690343584362473
[[24649 12247]
 [   54   217]]
balanced accuracy: 0.7344025032563856
```

Next:

- MinMaxScaler was used instead of StandardScaler just as a test, but the difference was not very noticeable, and StandardScaler often performed slightly better.
- PCA was used with n_componenets = 7, as it returned the best model accuracy, however PCA proved to be problematic in our Kaggle results
- When testing various final estimators, linear SVMs proved to be best (W/ C=10.0)

Results:

We get a model accuracy of **~70.431%** and a balanced accuracy of **~75.034%**

However, our Kaggle score was significantly worse: **~67.227%**

- Performance jumped the second PCA was removed

In [76]:
```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import KNNImputer

# Define preprocessor
preprocessor = Pipeline(
    [("scaler", MinMaxScaler()),
     ("selector", SelectFromModel(RandomForestClassifier(n_estimators=100)))]
)

# Create pipeline
pipe = Pipeline(
    [("preprocessor", preprocessor),
     ("pca", PCA(n_components=7)),
    ("model", StackingClassifier(estimators=base_models, final_estimator=SVC(kernel='li
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "adp_dist", "axon_dendrite_dist", "nucleus_dist", "rf_d
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.7043075846853392
[[25961 10935]
 [   55   216]]
balanced accuracy: 0.7503371899232869
```

Here, we have some significant improvement in performance.

All aforementioned features that we engineered (apart from the compartment features) are being used to test and train the algorithm.

- SelectFromModel was used to select features based on importance weights
  - This was done through Random Forest to estimate feature importance
  - The threshold is set to 0.8*mean, so features whose importance is greater than this threshold will be kept
- The base models were chosen through trial and error.
  - *Note: There was tuning done for the base models, but I had more success manually adding and removing various models*
- After getting rid of n_components for PCA, our performance improved by not specifying
- The final estimator was kept at linear SVM

Results:

We get a model accuracy of **~72.266%** and a balanced accuracy of **~78.156%**

Our Kaggle score improved significantly: **~73.467%**

*Note: This score actually ended up being my second highest privatel leaderboard score: **78.133%**.*

In [94]:
```
# Define preprocessor
# Scales/normalizes the data, then selects the most important features through Random Fo
preprocessor = Pipeline(
    [("scaler", MinMaxScaler()),
     ("selector", SelectFromModel(RandomForestClassifier(n_estimators=100), threshold='0
)

# define base models
# The models used for model stacking
base_models = [
    ("svc", SVC(kernel='linear', C=10.0, probability=True, random_state=2)),
    ("ada", AdaBoostClassifier(n_estimators=50, learning_rate=1.0, random_state=2)),
    ("knn", KNeighborsClassifier(n_neighbors=1, weights='uniform', algorithm='auto', lea
    ("rf", RandomForestClassifier(random_state=1, n_estimators=10, bootstrap=True)),
    ("xgb", XGBClassifier(random_state=2, max_depth=7, learning_rate=0.2, n_estimators=1
]

# create pipeline
pipe = Pipeline(
    [("preprocessor", preprocessor),
     ("pca", PCA()),
     ("model", StackingClassifier(estimators=base_models, final_estimator=SVC(kernel='li
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "fw_distance", "compartment_synapse_prop", "axon_dendri
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "fw_distance", "compa

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
accuracy: 0.7226571958995883
[[26631 10265]
 [   43   228]]
balanced accuracy: 0.7815569863471835
```

There were a number of base models that were tested for best performance

In [18]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import StackingClassifier, AdaBoostClassifier


# define base models
base_models1 = [
    ("lr", LogisticRegression()),
    ("knn", KNeighborsClassifier()),
    ("cart", DecisionTreeClassifier()),
    ("rf", RandomForestClassifier(random_state=1)),
    ("ada", AdaBoostClassifier(random_state=2)),
    ("svc", SVC(kernel='linear', C=10.0, probability=True, random_state=2))
]

base_models2 = [
    ("svc", SVC(kernel='linear', C=10.0, probability=True, random_state=2)),
    ("rf", RandomForestClassifier(random_state=1)),
    ("knn", KNeighborsClassifier()),
    ("ada", AdaBoostClassifier(random_state=2))
]

# define meta models
meta_model1 = LogisticRegression()
meta_model2 = LogisticRegression()

# define a list of models
models = [
    StackingClassifier(estimators=base_models1, final_estimator=meta_model1),
    StackingClassifier(estimators=base_models2, final_estimator=meta_model2),
]

# for each model
for model in models:
    # create pipeline
    pipe = Pipeline(
        [("scaler", StandardScaler()), ("model", model)]
    )

    # fit model
    pipe.fit(X_resampled, y_resampled)

    # perform cross-validation and print the average score
    scores = cross_val_score(pipe, X_resampled, y_resampled, cv=5, scoring='balanced_acc
    print(f"Average cross-validation score: {scores.mean()}")
```

```
Average cross-validation score: 0.7333333333333333
Average cross-validation score: 0.7310502283105021
```

Deep Learning

The last modeling technique used was deep learning, more specifically, feedforward neural networks

- The first attempt was a five-layer neural network with the Adam optimizer and cross entropy loss.

<u>The accuracies ended up being subpar, and were not tested in Kaggle:</u>

We get a model accuracy of **~63.928%** and a balanced accuracy of **~68.095%**

Note: If there was more time, I would've worked more on tuning the neural networks in terms of number of layers and neurons per layer.

In [55]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier

# Function to create model, required for KerasClassifier
def create_model():
    model = Sequential()
    model.add(Dense(20, input_dim=13, activation='relu'))  # Adjust input_dim to match n
    model.add(Dense(40, activation='relu'))
    model.add(Dense(60, activation='relu'))
    model.add(Dense(40, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

# Create model
model = KerasClassifier(build_fn=create_model, epochs=100, batch_size=10, verbose=0)

# Create pipeline
pipe = Pipeline([("scaler", StandardScaler()), ("model", model)])

# Fit model
pipe.fit(X_resampled, y_resampled)

# Predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "adp_dist", "axon_den

# Compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# Confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# Compute balanced accuracy
print(f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['p
```

```
C:\Users\micha\AppData\Local\Temp\ipykernel_20776\1989297797.py:20: DeprecationWarning:
KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) inst
ead. See https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
  model = KerasClassifier(build_fn=create_model, epochs=100, batch_size=10, verbose=0)
accuracy: 0.6392767777867463
[[23564 13332]
 [   75   196]]
balanced accuracy: 0.6809536249091892
```

Deep Learning Proved to be Difficult and Misleading

Below is the best performing FFNN that was possible.

- Three layers only, the input layer having 512 neurons and the hidden layer having only 16.
- Dropout was used to prevent overfitting, although it did not help as much as expected

---

<u>Results:</u>

We get a model accuracy of **~75.147%** and a balanced accuracy of **~75.212%**

Our Kaggle score was less than ideal: **~73.832%**

- The use of 300 epochs most likely led to overfitting

Nonetheless, this model was the best scoring model on Kaggle

```
In [67]: from sklearn.base import BaseEstimator, ClassifierMixin
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Dropout
         from tensorflow.keras.optimizers import Adam
         from sklearn.preprocessing import StandardScaler
         from sklearn.pipeline import Pipeline
         from imblearn.under_sampling import RandomUnderSampler
         import tensorflow as tf
         import numpy as np

         np.random.seed(0)
         tf.random.set_seed(0)

         class KerasClassifier(BaseEstimator, ClassifierMixin):
             def __init__(self, input_dim):
                 self.input_dim = input_dim
                 self.model = self._build_model()

             def _build_model(self):
                 model = Sequential()
                 model.add(Dense(512, input_dim=self.input_dim, activation='relu'))  # Input laye
                 model.add(Dropout(0.5))
                 model.add(Dense(16, activation='relu'))  # Hidden layer
                 model.add(Dropout(0.5))
                 model.add(Dense(1, activation='sigmoid'))  # Output layer
                 model.compile(loss='binary_crossentropy', optimizer=Adam(), metrics=['accuracy']
                 return model

             def fit(self, X, y):
                 self.model.fit(X, y, epochs=300, batch_size=5)
                 return self

             def predict_proba(self, X):
                 return self.model.predict(X)

         # create pipeline
         pipe = Pipeline(
             [("scaler", StandardScaler()),
              ("model", KerasClassifier(input_dim=X_resampled.shape[1]))]
         )

         # undersample majority class
         rus = RandomUnderSampler(random_state=0)
         X_resampled, y_resampled = rus.fit_resample(
             train_data[["fw_similarity", "fw_distance", "compartment_synapse_prop", "axon_dendri
             "compartment_apical",
             "compartment_apical_shaft",
             "compartment_apical_tuft",
             "compartment_axon",
             "compartment_basal",
             "compartment_oblique",
             "compartment_soma",
             "pre_apical_count",
             "pre_apical_shaft_count",
             "pre_apical_tuft_count",
             "pre_axon_count",
             "pre_basal_count",
             "pre_oblique_count",
             "pre_soma_count",
             "post_apical_count",
             "post_apical_shaft_count",
             "post_apical_tuft_count",
             "post_axon_count",
             "post_basal_count",
             "post_oblique_count",
             "post_soma_count"]],
             train_data["connected"]
         )

         # fit model
         pipe.fit(X_resampled, y_resampled)

         # predict on test data
         test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "fw_distance", "compa
             "compartment_apical",
             "compartment_apical_shaft",
             "compartment_apical_tuft",
             "compartment_axon",
             "compartment_basal",
             "compartment_oblique",
             "compartment_soma",
             "pre_apical_count",
             "pre_apical_shaft_count",
             "pre_apical_tuft_count",
             "pre_axon_count",
             "pre_basal_count",
             "pre_oblique_count",
             "pre_soma_count",
             "post_apical_count",
             "post_apical_shaft_count",
             "post_apical_tuft_count",
             "post_axon_count",
             "post_basal_count",
             "post_oblique_count",
             "post_soma_count"]])

         # compute accuracy
         print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

         # confusion matrix
         print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

         # compute balanced accuracy
         print(
             f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
         )
```

```
Epoch 1/300
438/438 [==============================] - 1s 2ms/step - loss: 0.5657 - accuracy: 0.7151
Epoch 2/300
438/438 [==============================] - 1s 2ms/step - loss: 0.5117 - accuracy: 0.7639
Epoch 3/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4983 - accuracy: 0.7612
Epoch 4/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4928 - accuracy: 0.7671
Epoch 5/300
438/438 [==============================] - 1s 3ms/step - loss: 0.4878 - accuracy: 0.7763
Epoch 6/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4706 - accuracy: 0.7804
Epoch 7/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4733 - accuracy: 0.7744
Epoch 8/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4597 - accuracy: 0.7795
Epoch 9/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4600 - accuracy: 0.7772
Epoch 10/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4526 - accuracy: 0.7808
Epoch 11/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4496 - accuracy: 0.7776
Epoch 12/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4529 - accuracy: 0.7890
```

```
Epoch 13/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4434 - accuracy: 0.7826
Epoch 14/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4435 - accuracy: 0.7900
Epoch 15/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4427 - accuracy: 0.7854
Epoch 16/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4399 - accuracy: 0.7854
Epoch 17/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4412 - accuracy: 0.7826
Epoch 18/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4320 - accuracy: 0.7904
Epoch 19/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4209 - accuracy: 0.8000
Epoch 20/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4311 - accuracy: 0.7909
Epoch 21/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4124 - accuracy: 0.7941
Epoch 22/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4220 - accuracy: 0.8027
Epoch 23/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4119 - accuracy: 0.7982
Epoch 24/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4025 - accuracy: 0.7977
Epoch 25/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3930 - accuracy: 0.8110
Epoch 26/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4078 - accuracy: 0.7877
Epoch 27/300
438/438 [==============================] - 1s 2ms/step - loss: 0.4007 - accuracy: 0.8064
Epoch 28/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3838 - accuracy: 0.8100
Epoch 29/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3978 - accuracy: 0.8059
Epoch 30/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3887 - accuracy: 0.8032
Epoch 31/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3917 - accuracy: 0.8100
Epoch 32/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3934 - accuracy: 0.8027
Epoch 33/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3734 - accuracy: 0.8151
Epoch 34/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3824 - accuracy: 0.8114
Epoch 35/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3881 - accuracy: 0.8110
Epoch 36/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3763 - accuracy: 0.8187
Epoch 37/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3601 - accuracy: 0.8169
Epoch 38/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3668 - accuracy: 0.8247
Epoch 39/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3780 - accuracy: 0.8091
Epoch 40/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3547 - accuracy: 0.8215
Epoch 41/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3582 - accuracy: 0.8174
Epoch 42/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3563 - accuracy: 0.8196
Epoch 43/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3621 - accuracy: 0.8201
Epoch 44/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3647 - accuracy: 0.8196
Epoch 45/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3618 - accuracy: 0.8247
```

```
Epoch 46/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3409 - accuracy: 0.8297
Epoch 47/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3609 - accuracy: 0.8192
Epoch 48/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3490 - accuracy: 0.8247
Epoch 49/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3464 - accuracy: 0.8242
Epoch 50/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3586 - accuracy: 0.8233
Epoch 51/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3320 - accuracy: 0.8274
Epoch 52/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3434 - accuracy: 0.8215
Epoch 53/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3391 - accuracy: 0.8370
Epoch 54/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3375 - accuracy: 0.8237
Epoch 55/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3167 - accuracy: 0.8311
Epoch 56/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3242 - accuracy: 0.8347
Epoch 57/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3345 - accuracy: 0.8320
Epoch 58/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3391 - accuracy: 0.8342
Epoch 59/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3278 - accuracy: 0.8320
Epoch 60/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3327 - accuracy: 0.8288
Epoch 61/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3171 - accuracy: 0.8342
Epoch 62/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3164 - accuracy: 0.8338
Epoch 63/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3133 - accuracy: 0.8361
Epoch 64/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3147 - accuracy: 0.8347
Epoch 65/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3242 - accuracy: 0.8324
Epoch 66/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3067 - accuracy: 0.8420
Epoch 67/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3208 - accuracy: 0.8397
Epoch 68/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3110 - accuracy: 0.8292
Epoch 69/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3152 - accuracy: 0.8347
Epoch 70/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3017 - accuracy: 0.8402
Epoch 71/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3075 - accuracy: 0.8347
Epoch 72/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3114 - accuracy: 0.8379
Epoch 73/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3041 - accuracy: 0.8425
Epoch 74/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3068 - accuracy: 0.8425
Epoch 75/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3055 - accuracy: 0.8420
Epoch 76/300
438/438 [==============================] - 1s 2ms/step - loss: 0.3153 - accuracy: 0.8361
Epoch 77/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2929 - accuracy: 0.8493
Epoch 78/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2935 - accuracy: 0.8393
```

```
Epoch 79/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2914 - accuracy: 0.8420
Epoch 80/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2877 - accuracy: 0.8438
Epoch 81/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2906 - accuracy: 0.8416
Epoch 82/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2980 - accuracy: 0.8411
Epoch 83/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2813 - accuracy: 0.8411
Epoch 84/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2887 - accuracy: 0.8516
Epoch 85/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2955 - accuracy: 0.8438
Epoch 86/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2979 - accuracy: 0.8443
Epoch 87/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2960 - accuracy: 0.8397
Epoch 88/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2771 - accuracy: 0.8493
Epoch 89/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2820 - accuracy: 0.8475
Epoch 90/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2887 - accuracy: 0.8470
Epoch 91/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2846 - accuracy: 0.8447
Epoch 92/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2798 - accuracy: 0.8521
Epoch 93/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2870 - accuracy: 0.8461
Epoch 94/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2743 - accuracy: 0.8479
Epoch 95/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2749 - accuracy: 0.8484
Epoch 96/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2796 - accuracy: 0.8479
Epoch 97/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2788 - accuracy: 0.8461
Epoch 98/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2999 - accuracy: 0.8420
Epoch 99/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2925 - accuracy: 0.8452
Epoch 100/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2810 - accuracy: 0.8507
Epoch 101/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2651 - accuracy: 0.8562
Epoch 102/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2676 - accuracy: 0.8594
Epoch 103/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2874 - accuracy: 0.8461
Epoch 104/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2606 - accuracy: 0.8502
Epoch 105/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2563 - accuracy: 0.8662
Epoch 106/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2740 - accuracy: 0.8612
Epoch 107/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2720 - accuracy: 0.8566
Epoch 108/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2783 - accuracy: 0.8498
Epoch 109/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2559 - accuracy: 0.8685
Epoch 110/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2758 - accuracy: 0.8689
Epoch 111/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2543 - accuracy: 0.8621
```

```
Epoch 112/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2660 - accuracy: 0.8607
Epoch 113/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2630 - accuracy: 0.8676
Epoch 114/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2489 - accuracy: 0.8689
Epoch 115/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2454 - accuracy: 0.8822
Epoch 116/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2322 - accuracy: 0.8877
Epoch 117/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2648 - accuracy: 0.8744
Epoch 118/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2540 - accuracy: 0.8749
Epoch 119/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2439 - accuracy: 0.8749
Epoch 120/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2492 - accuracy: 0.8749
Epoch 121/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2638 - accuracy: 0.8785
Epoch 122/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2517 - accuracy: 0.8831
Epoch 123/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2371 - accuracy: 0.8758
Epoch 124/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2550 - accuracy: 0.8799
Epoch 125/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2429 - accuracy: 0.8740
Epoch 126/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2433 - accuracy: 0.8831
Epoch 127/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2621 - accuracy: 0.8717
Epoch 128/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2412 - accuracy: 0.8858
Epoch 129/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2445 - accuracy: 0.8858
Epoch 130/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2358 - accuracy: 0.8826
Epoch 131/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2510 - accuracy: 0.8772
Epoch 132/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2421 - accuracy: 0.8826
Epoch 133/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2399 - accuracy: 0.8868
Epoch 134/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2361 - accuracy: 0.8849
Epoch 135/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2391 - accuracy: 0.8890
Epoch 136/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2515 - accuracy: 0.8799
Epoch 137/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2254 - accuracy: 0.8868
Epoch 138/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2400 - accuracy: 0.8877
Epoch 139/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2384 - accuracy: 0.8890
Epoch 140/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2390 - accuracy: 0.8845
Epoch 141/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2476 - accuracy: 0.8813
Epoch 142/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2240 - accuracy: 0.8890
Epoch 143/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2374 - accuracy: 0.8941
Epoch 144/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2441 - accuracy: 0.8890
```

```
Epoch 145/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2190 - accuracy: 0.8968
Epoch 146/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2169 - accuracy: 0.8927
Epoch 147/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2334 - accuracy: 0.9023
Epoch 148/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2219 - accuracy: 0.8963
Epoch 149/300
438/438 [==============================] - 1s 3ms/step - loss: 0.2196 - accuracy: 0.8936
Epoch 150/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2399 - accuracy: 0.8959
Epoch 151/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2252 - accuracy: 0.8959
Epoch 152/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2172 - accuracy: 0.8973
Epoch 153/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2376 - accuracy: 0.8936
Epoch 154/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2184 - accuracy: 0.9000
Epoch 155/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2138 - accuracy: 0.9018
Epoch 156/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2205 - accuracy: 0.8982
Epoch 157/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2203 - accuracy: 0.8941
Epoch 158/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2047 - accuracy: 0.9087
Epoch 159/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2155 - accuracy: 0.8959
Epoch 160/300
438/438 [==============================] - 1s 3ms/step - loss: 0.2088 - accuracy: 0.9041
Epoch 161/300
438/438 [==============================] - 1s 3ms/step - loss: 0.2177 - accuracy: 0.9005
Epoch 162/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2127 - accuracy: 0.9027
Epoch 163/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2123 - accuracy: 0.8918
Epoch 164/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2238 - accuracy: 0.8941
Epoch 165/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2055 - accuracy: 0.8963
Epoch 166/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2117 - accuracy: 0.9009
Epoch 167/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2045 - accuracy: 0.9055
Epoch 168/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2023 - accuracy: 0.9014
Epoch 169/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2128 - accuracy: 0.9046
Epoch 170/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2065 - accuracy: 0.9068
Epoch 171/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2162 - accuracy: 0.9014
Epoch 172/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2135 - accuracy: 0.9032
Epoch 173/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2094 - accuracy: 0.9009
Epoch 174/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2191 - accuracy: 0.9009
Epoch 175/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2096 - accuracy: 0.9087
Epoch 176/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2137 - accuracy: 0.9105
Epoch 177/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1934 - accuracy: 0.9100
```

```
Epoch 178/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2064 - accuracy: 0.9023
Epoch 179/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2103 - accuracy: 0.9050
Epoch 180/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2085 - accuracy: 0.9100
Epoch 181/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1916 - accuracy: 0.9128
Epoch 182/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2181 - accuracy: 0.9078
Epoch 183/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1902 - accuracy: 0.9114
Epoch 184/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2091 - accuracy: 0.9073
Epoch 185/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2071 - accuracy: 0.9091
Epoch 186/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2007 - accuracy: 0.9100
Epoch 187/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1946 - accuracy: 0.9155
Epoch 188/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1885 - accuracy: 0.9228
Epoch 189/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1846 - accuracy: 0.9183
Epoch 190/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2084 - accuracy: 0.9105
Epoch 191/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1959 - accuracy: 0.9119
Epoch 192/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1971 - accuracy: 0.9100
Epoch 193/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2010 - accuracy: 0.9142
Epoch 194/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1875 - accuracy: 0.9174
Epoch 195/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1871 - accuracy: 0.9183
Epoch 196/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1854 - accuracy: 0.9215
Epoch 197/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1981 - accuracy: 0.9178
Epoch 198/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1994 - accuracy: 0.9110
Epoch 199/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2006 - accuracy: 0.9137
Epoch 200/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1939 - accuracy: 0.9192
Epoch 201/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1916 - accuracy: 0.9178
Epoch 202/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1923 - accuracy: 0.9219
Epoch 203/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2062 - accuracy: 0.9169
Epoch 204/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1852 - accuracy: 0.9196
Epoch 205/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1907 - accuracy: 0.9174
Epoch 206/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1778 - accuracy: 0.9196
Epoch 207/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1972 - accuracy: 0.9183
Epoch 208/300
438/438 [==============================] - 1s 2ms/step - loss: 0.2139 - accuracy: 0.9201
Epoch 209/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1973 - accuracy: 0.9187
Epoch 210/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1851 - accuracy: 0.9205
```

```
Epoch 211/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1962 - accuracy: 0.9155
Epoch 212/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1827 - accuracy: 0.9301
Epoch 213/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1723 - accuracy: 0.9210
Epoch 214/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1817 - accuracy: 0.9279
Epoch 215/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1706 - accuracy: 0.9205
Epoch 216/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1733 - accuracy: 0.9242
Epoch 217/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1791 - accuracy: 0.9301
Epoch 218/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1944 - accuracy: 0.9068
Epoch 219/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1928 - accuracy: 0.9183
Epoch 220/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1979 - accuracy: 0.9164
Epoch 221/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1876 - accuracy: 0.9201
Epoch 222/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1635 - accuracy: 0.9288
Epoch 223/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1706 - accuracy: 0.9274
Epoch 224/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1802 - accuracy: 0.9215
Epoch 225/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1746 - accuracy: 0.9315
Epoch 226/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1986 - accuracy: 0.9237
Epoch 227/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1869 - accuracy: 0.9251
Epoch 228/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1880 - accuracy: 0.9279
Epoch 229/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1726 - accuracy: 0.9269
Epoch 230/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1597 - accuracy: 0.9329
Epoch 231/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1773 - accuracy: 0.9306
Epoch 232/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1813 - accuracy: 0.9320
Epoch 233/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1948 - accuracy: 0.9192
Epoch 234/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1628 - accuracy: 0.9301
Epoch 235/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1793 - accuracy: 0.9269
Epoch 236/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1815 - accuracy: 0.9224
Epoch 237/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1814 - accuracy: 0.9292
Epoch 238/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1719 - accuracy: 0.9242
Epoch 239/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1842 - accuracy: 0.9301
Epoch 240/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1678 - accuracy: 0.9342
Epoch 241/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1812 - accuracy: 0.9233
Epoch 242/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1781 - accuracy: 0.9201
Epoch 243/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1588 - accuracy: 0.9301
```

```
Epoch 244/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1763 - accuracy: 0.9306
Epoch 245/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1573 - accuracy: 0.9361
Epoch 246/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1702 - accuracy: 0.9342
Epoch 247/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1579 - accuracy: 0.9342
Epoch 248/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1601 - accuracy: 0.9374
Epoch 249/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1793 - accuracy: 0.9288
Epoch 250/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1746 - accuracy: 0.9347
Epoch 251/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1693 - accuracy: 0.9320
Epoch 252/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1503 - accuracy: 0.9416
Epoch 253/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1612 - accuracy: 0.9361
Epoch 254/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1702 - accuracy: 0.9247
Epoch 255/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1684 - accuracy: 0.9320
Epoch 256/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1687 - accuracy: 0.9251
Epoch 257/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1635 - accuracy: 0.9374
Epoch 258/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1680 - accuracy: 0.9324
Epoch 259/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1577 - accuracy: 0.9324
Epoch 260/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1728 - accuracy: 0.9347
Epoch 261/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1534 - accuracy: 0.9333
Epoch 262/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1494 - accuracy: 0.9384
Epoch 263/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1606 - accuracy: 0.9411
Epoch 264/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1536 - accuracy: 0.9374
Epoch 265/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1560 - accuracy: 0.9283
Epoch 266/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1765 - accuracy: 0.9370
Epoch 267/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1750 - accuracy: 0.9338
Epoch 268/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1589 - accuracy: 0.9324
Epoch 269/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1636 - accuracy: 0.9402
Epoch 270/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1527 - accuracy: 0.9301
Epoch 271/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1525 - accuracy: 0.9342
Epoch 272/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1645 - accuracy: 0.9324
Epoch 273/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1605 - accuracy: 0.9365
Epoch 274/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1637 - accuracy: 0.9338
Epoch 275/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1557 - accuracy: 0.9438
Epoch 276/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1595 - accuracy: 0.9297
```

```
Epoch 277/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1649 - accuracy: 0.9370
Epoch 278/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1492 - accuracy: 0.9374
Epoch 279/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1653 - accuracy: 0.9342
Epoch 280/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1385 - accuracy: 0.9425
Epoch 281/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1492 - accuracy: 0.9429
Epoch 282/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1499 - accuracy: 0.9388
Epoch 283/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1818 - accuracy: 0.9320
Epoch 284/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1549 - accuracy: 0.9406
Epoch 285/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1556 - accuracy: 0.9342
Epoch 286/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1600 - accuracy: 0.9397
Epoch 287/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1464 - accuracy: 0.9384
Epoch 288/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1334 - accuracy: 0.9438
Epoch 289/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1534 - accuracy: 0.9416
Epoch 290/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1596 - accuracy: 0.9374
Epoch 291/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1463 - accuracy: 0.9416
Epoch 292/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1700 - accuracy: 0.9347
Epoch 293/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1504 - accuracy: 0.9425
Epoch 294/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1608 - accuracy: 0.9388
Epoch 295/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1878 - accuracy: 0.9342
Epoch 296/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1476 - accuracy: 0.9397
Epoch 297/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1331 - accuracy: 0.9420
Epoch 298/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1401 - accuracy: 0.9466
Epoch 299/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1423 - accuracy: 0.9443
Epoch 300/300
438/438 [==============================] - 1s 2ms/step - loss: 0.1461 - accuracy: 0.9393
accuracy: 0.7514730809589152
[[27726  9170]
 [   67   204]]
balanced accuracy: 0.752115550481177
```

Cross-Validation

In [68]:
```python
import numpy as np
from sklearn.model_selection import KFold

class KerasClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, input_dim):
        self.input_dim = input_dim
        self.model = self._build_model()

    def _build_model(self):
        model = Sequential()
        model.add(Dense(512, input_dim=self.input_dim, activation='relu'))  # Input laye
        model.add(Dropout(0.5))
        model.add(Dense(16, activation='relu'))  # Hidden layer
        model.add(Dropout(0.5))
        model.add(Dense(1, activation='sigmoid'))  # Output layer
        model.compile(loss='binary_crossentropy', optimizer=Adam(), metrics=['accuracy']
        return model

    def fit(self, X, y):
        self.model.fit(X, y, epochs=300, batch_size=5)
        return self

    def predict_proba(self, X):
        return self.model.predict(X)

    def predict(self, X):
        return (self.predict_proba(X) > 0.5).astype(int)

# Define the number of folds for cross-validation
n_folds = 5
kfold = KFold(n_splits=n_folds, shuffle=True)

# Initialize list to store scores for each fold
scores = []

# Loop over each fold
for train, test in kfold.split(X_resampled, y_resampled):
    # Reset the pipeline
    pipe = Pipeline(
        [("scaler", StandardScaler()),
         ("model", KerasClassifier(input_dim=X_resampled.shape[1]))]
    )

    # Select the training and testing data for this fold
    X_train, X_test = X_resampled.iloc[train], X_resampled.iloc[test]
    y_train, y_test = y_resampled.iloc[train], y_resampled.iloc[test]

    # Fit the model
    pipe.fit(X_train, y_train)

    # Evaluate the model
    score = pipe.score(X_test, y_test)

    # Store the score
    scores.append(score)

# Print the mean and standard deviation of the scores
print(f"Cross-validation mean accuracy: {np.mean(scores)}, std: {np.std(scores)}")
```

```
Epoch 1/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5650 - accuracy: 0.7203
Epoch 2/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5073 - accuracy: 0.7643
Epoch 3/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4950 - accuracy: 0.7717
Epoch 4/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4898 - accuracy: 0.7694
Epoch 5/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4757 - accuracy: 0.7791
Epoch 6/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4845 - accuracy: 0.7631
Epoch 7/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4737 - accuracy: 0.7717
Epoch 8/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4667 - accuracy: 0.7734
Epoch 9/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.4634 - accuracy: 0.7757
Epoch 10/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4440 - accuracy: 0.7939
Epoch 11/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4593 - accuracy: 0.7882
Epoch 12/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4412 - accuracy: 0.7900
Epoch 13/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4658 - accuracy: 0.7825
Epoch 14/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4413 - accuracy: 0.7934
Epoch 15/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4249 - accuracy: 0.7945
Epoch 16/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4281 - accuracy: 0.7991
Epoch 17/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4288 - accuracy: 0.7991
Epoch 18/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4157 - accuracy: 0.7997
Epoch 19/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4117 - accuracy: 0.8019
Epoch 20/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4145 - accuracy: 0.7900
Epoch 21/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4033 - accuracy: 0.8002
Epoch 22/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4033 - accuracy: 0.8054
Epoch 23/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4144 - accuracy: 0.7939
Epoch 24/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4065 - accuracy: 0.8037
Epoch 25/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4086 - accuracy: 0.7957
Epoch 26/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3998 - accuracy: 0.8065
Epoch 27/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3849 - accuracy: 0.8076
Epoch 28/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3829 - accuracy: 0.8082
Epoch 29/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3850 - accuracy: 0.8116
Epoch 30/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3793 - accuracy: 0.8145
Epoch 31/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3825 - accuracy: 0.8105
Epoch 32/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3682 - accuracy: 0.8242
Epoch 33/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3929 - accuracy: 0.8111
Epoch 34/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3693 - accuracy: 0.8168
Epoch 35/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3775 - accuracy: 0.8151
Epoch 36/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3779 - accuracy: 0.8179
Epoch 37/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3640 - accuracy: 0.8162
Epoch 38/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3609 - accuracy: 0.8196
Epoch 39/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3537 - accuracy: 0.8236
Epoch 40/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3592 - accuracy: 0.8145
Epoch 41/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3534 - accuracy: 0.8248
Epoch 42/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.3610 - accuracy: 0.8219
Epoch 43/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3730 - accuracy: 0.8213
Epoch 44/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3515 - accuracy: 0.8253
Epoch 45/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3518 - accuracy: 0.8253
Epoch 46/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3451 - accuracy: 0.8259
Epoch 47/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3278 - accuracy: 0.8322
Epoch 48/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3577 - accuracy: 0.8225
Epoch 49/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3532 - accuracy: 0.8253
Epoch 50/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3382 - accuracy: 0.8259
Epoch 51/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3553 - accuracy: 0.8253
Epoch 52/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3212 - accuracy: 0.8379
Epoch 53/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3462 - accuracy: 0.8316
Epoch 54/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3357 - accuracy: 0.8316
Epoch 55/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3391 - accuracy: 0.8333
Epoch 56/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3371 - accuracy: 0.8293
Epoch 57/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3223 - accuracy: 0.8436
Epoch 58/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3301 - accuracy: 0.8293
Epoch 59/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3323 - accuracy: 0.8385
Epoch 60/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3211 - accuracy: 0.8425
Epoch 61/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3379 - accuracy: 0.8379
Epoch 62/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3172 - accuracy: 0.8436
Epoch 63/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3198 - accuracy: 0.8333
Epoch 64/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3174 - accuracy: 0.8419
Epoch 65/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3090 - accuracy: 0.8368
Epoch 66/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3379 - accuracy: 0.8305
Epoch 67/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3170 - accuracy: 0.8430
Epoch 68/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3149 - accuracy: 0.8385
Epoch 69/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3010 - accuracy: 0.8453
Epoch 70/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3082 - accuracy: 0.8493
Epoch 71/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3222 - accuracy: 0.8482
Epoch 72/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3004 - accuracy: 0.8522
Epoch 73/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3130 - accuracy: 0.8408
Epoch 74/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2993 - accuracy: 0.8545
Epoch 75/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2987 - accuracy: 0.8493
Epoch 76/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3034 - accuracy: 0.8505
Epoch 77/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2847 - accuracy: 0.8505
Epoch 78/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2930 - accuracy: 0.8522
Epoch 79/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2929 - accuracy: 0.8476
Epoch 80/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2934 - accuracy: 0.8465
Epoch 81/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3008 - accuracy: 0.8533
Epoch 82/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2843 - accuracy: 0.8562
Epoch 83/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2895 - accuracy: 0.8562
Epoch 84/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3035 - accuracy: 0.8584
Epoch 85/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2902 - accuracy: 0.8499
Epoch 86/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2890 - accuracy: 0.8533
Epoch 87/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2819 - accuracy: 0.8630
Epoch 88/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2853 - accuracy: 0.8596
Epoch 89/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2929 - accuracy: 0.8527
Epoch 90/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2814 - accuracy: 0.8602
Epoch 91/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2761 - accuracy: 0.8556
Epoch 92/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2836 - accuracy: 0.8602
Epoch 93/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2845 - accuracy: 0.8613
Epoch 94/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2943 - accuracy: 0.8567
Epoch 95/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2865 - accuracy: 0.8624
Epoch 96/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2810 - accuracy: 0.8624
Epoch 97/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2762 - accuracy: 0.8647
Epoch 98/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2705 - accuracy: 0.8676
Epoch 99/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2721 - accuracy: 0.8533
Epoch 100/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2758 - accuracy: 0.8630
Epoch 101/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2563 - accuracy: 0.8607
Epoch 102/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2689 - accuracy: 0.8613
Epoch 103/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2665 - accuracy: 0.8613
Epoch 104/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2813 - accuracy: 0.8670
Epoch 105/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2966 - accuracy: 0.8647
Epoch 106/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2570 - accuracy: 0.8664
Epoch 107/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2670 - accuracy: 0.8619
Epoch 108/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2590 - accuracy: 0.8659
Epoch 109/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2687 - accuracy: 0.8659
Epoch 110/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2684 - accuracy: 0.8642
Epoch 111/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2586 - accuracy: 0.8687
Epoch 112/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2595 - accuracy: 0.8653
Epoch 113/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2539 - accuracy: 0.8784
Epoch 114/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2647 - accuracy: 0.8636
Epoch 115/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2418 - accuracy: 0.8767
Epoch 116/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2653 - accuracy: 0.8721
Epoch 117/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2359 - accuracy: 0.8824
Epoch 118/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2882 - accuracy: 0.8682
Epoch 119/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2577 - accuracy: 0.8699
Epoch 120/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2483 - accuracy: 0.8756
Epoch 121/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2555 - accuracy: 0.8670
Epoch 122/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2416 - accuracy: 0.8693
Epoch 123/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2673 - accuracy: 0.8699
Epoch 124/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2475 - accuracy: 0.8733
Epoch 125/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2493 - accuracy: 0.8670
Epoch 126/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2446 - accuracy: 0.8716
Epoch 127/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2502 - accuracy: 0.8693
Epoch 128/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2409 - accuracy: 0.8796
Epoch 129/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2305 - accuracy: 0.8761
Epoch 130/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2422 - accuracy: 0.8739
Epoch 131/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2573 - accuracy: 0.8739
Epoch 132/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2426 - accuracy: 0.8779
Epoch 133/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2217 - accuracy: 0.8864
Epoch 134/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2483 - accuracy: 0.8721
Epoch 135/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2276 - accuracy: 0.8824
Epoch 136/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2333 - accuracy: 0.8807
Epoch 137/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2448 - accuracy: 0.8773
Epoch 138/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2308 - accuracy: 0.8784
Epoch 139/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2237 - accuracy: 0.8824
Epoch 140/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2355 - accuracy: 0.8767
Epoch 141/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2316 - accuracy: 0.8790
Epoch 142/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2601 - accuracy: 0.8636
Epoch 143/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2469 - accuracy: 0.8767
Epoch 144/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2195 - accuracy: 0.8853
Epoch 145/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2337 - accuracy: 0.8773
Epoch 146/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2428 - accuracy: 0.8756
Epoch 147/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2348 - accuracy: 0.8767
Epoch 148/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2302 - accuracy: 0.8807
Epoch 149/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2470 - accuracy: 0.8773
Epoch 150/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2361 - accuracy: 0.8773
Epoch 151/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2293 - accuracy: 0.8898
Epoch 152/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2485 - accuracy: 0.8721
Epoch 153/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2285 - accuracy: 0.8898
Epoch 154/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2131 - accuracy: 0.8933
Epoch 155/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2318 - accuracy: 0.8870
Epoch 156/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2489 - accuracy: 0.8693
Epoch 157/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2356 - accuracy: 0.8750
Epoch 158/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2173 - accuracy: 0.8887
Epoch 159/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2316 - accuracy: 0.8807
Epoch 160/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2213 - accuracy: 0.8841
Epoch 161/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2241 - accuracy: 0.8858
Epoch 162/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2022 - accuracy: 0.8944
Epoch 163/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2195 - accuracy: 0.8881
Epoch 164/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2224 - accuracy: 0.8881
Epoch 165/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2363 - accuracy: 0.8858
Epoch 166/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2060 - accuracy: 0.8950
Epoch 167/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2094 - accuracy: 0.8887
Epoch 168/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2142 - accuracy: 0.8898
Epoch 169/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2264 - accuracy: 0.8836
Epoch 170/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2276 - accuracy: 0.8841
Epoch 171/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1965 - accuracy: 0.8955
Epoch 172/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2143 - accuracy: 0.8961
Epoch 173/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2196 - accuracy: 0.8887
Epoch 174/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2209 - accuracy: 0.8830
Epoch 175/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2112 - accuracy: 0.8841
Epoch 176/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2109 - accuracy: 0.8898
Epoch 177/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1947 - accuracy: 0.8933
Epoch 178/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2112 - accuracy: 0.8870
Epoch 179/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2112 - accuracy: 0.8916
Epoch 180/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2490 - accuracy: 0.8744
Epoch 181/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2073 - accuracy: 0.8887
Epoch 182/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2426 - accuracy: 0.8801
Epoch 183/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2170 - accuracy: 0.8876
Epoch 184/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1910 - accuracy: 0.8967
Epoch 185/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2140 - accuracy: 0.8887
Epoch 186/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2017 - accuracy: 0.8950
Epoch 187/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2089 - accuracy: 0.8887
Epoch 188/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1995 - accuracy: 0.8984
Epoch 189/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2281 - accuracy: 0.8853
Epoch 190/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2177 - accuracy: 0.8841
Epoch 191/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2130 - accuracy: 0.8904
Epoch 192/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2019 - accuracy: 0.8881
Epoch 193/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2054 - accuracy: 0.8955
Epoch 194/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2078 - accuracy: 0.8916
Epoch 195/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2125 - accuracy: 0.8910
Epoch 196/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2170 - accuracy: 0.8921
Epoch 197/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2114 - accuracy: 0.8916
Epoch 198/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1947 - accuracy: 0.8950
Epoch 199/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2091 - accuracy: 0.9030
Epoch 200/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1966 - accuracy: 0.8990
Epoch 201/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1856 - accuracy: 0.8967
Epoch 202/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2076 - accuracy: 0.8904
Epoch 203/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2074 - accuracy: 0.8961
Epoch 204/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2054 - accuracy: 0.8961
Epoch 205/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2079 - accuracy: 0.8916
Epoch 206/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2245 - accuracy: 0.8847
Epoch 207/300
```

```
351/351 [==============================] - 1s 3ms/step - loss: 0.2048 - accuracy: 0.8955
Epoch 208/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1955 - accuracy: 0.9007
Epoch 209/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1880 - accuracy: 0.9070
Epoch 210/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1949 - accuracy: 0.9007
Epoch 211/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1983 - accuracy: 0.8961
Epoch 212/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1975 - accuracy: 0.9018
Epoch 213/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2060 - accuracy: 0.8916
Epoch 214/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1867 - accuracy: 0.8995
Epoch 215/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1818 - accuracy: 0.8950
Epoch 216/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1898 - accuracy: 0.8944
Epoch 217/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2049 - accuracy: 0.9013
Epoch 218/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1898 - accuracy: 0.9001
Epoch 219/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1823 - accuracy: 0.8967
Epoch 220/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2007 - accuracy: 0.9007
Epoch 221/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1922 - accuracy: 0.8995
Epoch 222/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1913 - accuracy: 0.8904
Epoch 223/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1810 - accuracy: 0.9064
Epoch 224/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1822 - accuracy: 0.9075
Epoch 225/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1708 - accuracy: 0.9110
Epoch 226/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1744 - accuracy: 0.8995
Epoch 227/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1738 - accuracy: 0.9024
Epoch 228/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1868 - accuracy: 0.8961
Epoch 229/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1854 - accuracy: 0.9001
Epoch 230/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1746 - accuracy: 0.9035
Epoch 231/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1786 - accuracy: 0.9007
Epoch 232/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1902 - accuracy: 0.9064
Epoch 233/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1933 - accuracy: 0.9018
Epoch 234/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1746 - accuracy: 0.9081
Epoch 235/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1895 - accuracy: 0.8916
Epoch 236/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1869 - accuracy: 0.8961
Epoch 237/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1780 - accuracy: 0.8973
Epoch 238/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1902 - accuracy: 0.8927
Epoch 239/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1821 - accuracy: 0.8984
Epoch 240/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1803 - accuracy: 0.9018
Epoch 241/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1701 - accuracy: 0.9064
Epoch 242/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1828 - accuracy: 0.8984
Epoch 243/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1922 - accuracy: 0.8984
Epoch 244/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1786 - accuracy: 0.9047
Epoch 245/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1822 - accuracy: 0.9013
Epoch 246/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1893 - accuracy: 0.9081
Epoch 247/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1959 - accuracy: 0.8961
Epoch 248/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1714 - accuracy: 0.9075
Epoch 249/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1983 - accuracy: 0.9058
Epoch 250/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1654 - accuracy: 0.9104
Epoch 251/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1496 - accuracy: 0.9138
Epoch 252/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1709 - accuracy: 0.9070
Epoch 253/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1685 - accuracy: 0.9087
Epoch 254/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1876 - accuracy: 0.8990
Epoch 255/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1747 - accuracy: 0.9081
Epoch 256/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1846 - accuracy: 0.8995
Epoch 257/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2093 - accuracy: 0.9007
Epoch 258/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1555 - accuracy: 0.9132
Epoch 259/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1652 - accuracy: 0.9047
Epoch 260/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1775 - accuracy: 0.9115
Epoch 261/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1943 - accuracy: 0.9184
Epoch 262/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1789 - accuracy: 0.9121
Epoch 263/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1592 - accuracy: 0.9269
Epoch 264/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1928 - accuracy: 0.9132
Epoch 265/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1641 - accuracy: 0.9241
Epoch 266/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1610 - accuracy: 0.9201
Epoch 267/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1843 - accuracy: 0.9115
Epoch 268/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1647 - accuracy: 0.9098
Epoch 269/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1907 - accuracy: 0.9098
Epoch 270/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1597 - accuracy: 0.9247
Epoch 271/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1592 - accuracy: 0.9212
Epoch 272/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1542 - accuracy: 0.9207
Epoch 273/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1509 - accuracy: 0.9229
Epoch 274/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1721 - accuracy: 0.9081
Epoch 275/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1626 - accuracy: 0.9075
Epoch 276/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1621 - accuracy: 0.9127
Epoch 277/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1677 - accuracy: 0.9127
Epoch 278/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1768 - accuracy: 0.9127
Epoch 279/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1604 - accuracy: 0.9224
Epoch 280/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1578 - accuracy: 0.9195
Epoch 281/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1935 - accuracy: 0.9172
Epoch 282/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1631 - accuracy: 0.9207
Epoch 283/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1817 - accuracy: 0.9144
Epoch 284/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1619 - accuracy: 0.9247
Epoch 285/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1673 - accuracy: 0.9229
Epoch 286/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1660 - accuracy: 0.9212
Epoch 287/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1731 - accuracy: 0.9229
Epoch 288/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1484 - accuracy: 0.9275
Epoch 289/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1573 - accuracy: 0.9224
Epoch 290/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1615 - accuracy: 0.9275
Epoch 291/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1987 - accuracy: 0.9178
Epoch 292/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1706 - accuracy: 0.9258
Epoch 293/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1767 - accuracy: 0.9201
Epoch 294/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1789 - accuracy: 0.9201
Epoch 295/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1683 - accuracy: 0.9161
Epoch 296/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1523 - accuracy: 0.9338
Epoch 297/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1581 - accuracy: 0.9252
Epoch 298/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1438 - accuracy: 0.9366
Epoch 299/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1528 - accuracy: 0.9315
Epoch 300/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1976 - accuracy: 0.9315
Epoch 1/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5642 - accuracy: 0.7078
Epoch 2/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5138 - accuracy: 0.7586
Epoch 3/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5076 - accuracy: 0.7608
Epoch 4/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4845 - accuracy: 0.7677
Epoch 5/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4797 - accuracy: 0.7711
Epoch 6/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.4775 - accuracy: 0.7871
Epoch 7/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4659 - accuracy: 0.7848
Epoch 8/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4635 - accuracy: 0.7825
Epoch 9/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4564 - accuracy: 0.7831
Epoch 10/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4524 - accuracy: 0.7905
Epoch 11/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4438 - accuracy: 0.7911
Epoch 12/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4440 - accuracy: 0.7871
Epoch 13/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4411 - accuracy: 0.7871
Epoch 14/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4410 - accuracy: 0.7911
Epoch 15/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4425 - accuracy: 0.7979
Epoch 16/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4235 - accuracy: 0.7951
Epoch 17/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4190 - accuracy: 0.7968
Epoch 18/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4209 - accuracy: 0.7979
Epoch 19/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4059 - accuracy: 0.7985
Epoch 20/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4200 - accuracy: 0.8025
Epoch 21/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4174 - accuracy: 0.8031
Epoch 22/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4113 - accuracy: 0.7997
Epoch 23/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3958 - accuracy: 0.8065
Epoch 24/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4020 - accuracy: 0.8054
Epoch 25/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3951 - accuracy: 0.8099
Epoch 26/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3926 - accuracy: 0.8059
Epoch 27/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3914 - accuracy: 0.8174
Epoch 28/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3803 - accuracy: 0.8156
Epoch 29/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3770 - accuracy: 0.8134
Epoch 30/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3721 - accuracy: 0.8191
Epoch 31/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3779 - accuracy: 0.8162
Epoch 32/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3671 - accuracy: 0.8208
Epoch 33/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3789 - accuracy: 0.8219
Epoch 34/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3784 - accuracy: 0.8128
Epoch 35/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3828 - accuracy: 0.8151
Epoch 36/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3635 - accuracy: 0.8134
Epoch 37/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3570 - accuracy: 0.8265
Epoch 38/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3493 - accuracy: 0.8362
Epoch 39/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.3469 - accuracy: 0.8265
Epoch 40/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3582 - accuracy: 0.8248
Epoch 41/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3377 - accuracy: 0.8236
Epoch 42/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3516 - accuracy: 0.8213
Epoch 43/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3348 - accuracy: 0.8328
Epoch 44/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3396 - accuracy: 0.8265
Epoch 45/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3430 - accuracy: 0.8185
Epoch 46/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3335 - accuracy: 0.8345
Epoch 47/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3266 - accuracy: 0.8368
Epoch 48/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3396 - accuracy: 0.8333
Epoch 49/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3267 - accuracy: 0.8299
Epoch 50/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3439 - accuracy: 0.8271
Epoch 51/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3171 - accuracy: 0.8328
Epoch 52/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3080 - accuracy: 0.8390
Epoch 53/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3266 - accuracy: 0.8305
Epoch 54/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3173 - accuracy: 0.8413
Epoch 55/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3084 - accuracy: 0.8396
Epoch 56/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3236 - accuracy: 0.8430
Epoch 57/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3253 - accuracy: 0.8328
Epoch 58/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3224 - accuracy: 0.8390
Epoch 59/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2931 - accuracy: 0.8453
Epoch 60/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2884 - accuracy: 0.8527
Epoch 61/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3187 - accuracy: 0.8345
Epoch 62/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3068 - accuracy: 0.8453
Epoch 63/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3017 - accuracy: 0.8408
Epoch 64/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3045 - accuracy: 0.8442
Epoch 65/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3008 - accuracy: 0.8413
Epoch 66/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3052 - accuracy: 0.8459
Epoch 67/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3045 - accuracy: 0.8419
Epoch 68/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2797 - accuracy: 0.8459
Epoch 69/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2819 - accuracy: 0.8556
Epoch 70/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3083 - accuracy: 0.8442
Epoch 71/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3020 - accuracy: 0.8390
Epoch 72/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2837 - accuracy: 0.8442
Epoch 73/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2923 - accuracy: 0.8436
Epoch 74/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2772 - accuracy: 0.8533
Epoch 75/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3062 - accuracy: 0.8487
Epoch 76/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2751 - accuracy: 0.8493
Epoch 77/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2670 - accuracy: 0.8539
Epoch 78/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2598 - accuracy: 0.8550
Epoch 79/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2806 - accuracy: 0.8550
Epoch 80/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2814 - accuracy: 0.8550
Epoch 81/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2854 - accuracy: 0.8436
Epoch 82/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2669 - accuracy: 0.8459
Epoch 83/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2564 - accuracy: 0.8596
Epoch 84/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2633 - accuracy: 0.8584
Epoch 85/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2721 - accuracy: 0.8807
Epoch 86/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2693 - accuracy: 0.8630
Epoch 87/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2951 - accuracy: 0.8659
Epoch 88/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2744 - accuracy: 0.8670
Epoch 89/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2620 - accuracy: 0.8693
Epoch 90/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2549 - accuracy: 0.8676
Epoch 91/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2618 - accuracy: 0.8642
Epoch 92/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2738 - accuracy: 0.8727
Epoch 93/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2701 - accuracy: 0.8670
Epoch 94/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2450 - accuracy: 0.8721
Epoch 95/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2573 - accuracy: 0.8733
Epoch 96/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2599 - accuracy: 0.8756
Epoch 97/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2433 - accuracy: 0.8767
Epoch 98/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2563 - accuracy: 0.8704
Epoch 99/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2446 - accuracy: 0.8853
Epoch 100/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2669 - accuracy: 0.8733
Epoch 101/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2471 - accuracy: 0.8767
Epoch 102/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2461 - accuracy: 0.8767
Epoch 103/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2474 - accuracy: 0.8836
Epoch 104/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2355 - accuracy: 0.8841
Epoch 105/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2468 - accuracy: 0.8710
Epoch 106/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2418 - accuracy: 0.8739
Epoch 107/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2394 - accuracy: 0.8824
Epoch 108/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2556 - accuracy: 0.8824
Epoch 109/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2387 - accuracy: 0.8847
Epoch 110/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2235 - accuracy: 0.8921
Epoch 111/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2441 - accuracy: 0.8870
Epoch 112/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2219 - accuracy: 0.8973
Epoch 113/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2219 - accuracy: 0.8887
Epoch 114/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2390 - accuracy: 0.8921
Epoch 115/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2319 - accuracy: 0.8904
Epoch 116/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2431 - accuracy: 0.8887
Epoch 117/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2362 - accuracy: 0.8841
Epoch 118/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2329 - accuracy: 0.8950
Epoch 119/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2249 - accuracy: 0.8967
Epoch 120/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2379 - accuracy: 0.8938
Epoch 121/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2201 - accuracy: 0.8984
Epoch 122/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2477 - accuracy: 0.8853
Epoch 123/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2257 - accuracy: 0.9007
Epoch 124/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2187 - accuracy: 0.9064
Epoch 125/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2169 - accuracy: 0.9001
Epoch 126/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2263 - accuracy: 0.8984
Epoch 127/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2217 - accuracy: 0.9035
Epoch 128/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2280 - accuracy: 0.8973
Epoch 129/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2312 - accuracy: 0.8938
Epoch 130/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2131 - accuracy: 0.9018
Epoch 131/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2338 - accuracy: 0.8978
Epoch 132/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1952 - accuracy: 0.9058
Epoch 133/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2120 - accuracy: 0.9070
Epoch 134/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2144 - accuracy: 0.8967
Epoch 135/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2179 - accuracy: 0.8967
Epoch 136/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2195 - accuracy: 0.8944
Epoch 137/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2180 - accuracy: 0.9041
Epoch 138/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2108 - accuracy: 0.9041
Epoch 139/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2085 - accuracy: 0.8950
Epoch 140/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2141 - accuracy: 0.8990
Epoch 141/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2065 - accuracy: 0.9035
Epoch 142/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2152 - accuracy: 0.8961
Epoch 143/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2025 - accuracy: 0.9058
Epoch 144/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1971 - accuracy: 0.9127
Epoch 145/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1928 - accuracy: 0.9070
Epoch 146/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2280 - accuracy: 0.9007
Epoch 147/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2125 - accuracy: 0.9064
Epoch 148/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2045 - accuracy: 0.9070
Epoch 149/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2152 - accuracy: 0.9115
Epoch 150/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1996 - accuracy: 0.9081
Epoch 151/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2013 - accuracy: 0.9104
Epoch 152/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1906 - accuracy: 0.9178
Epoch 153/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2061 - accuracy: 0.9121
Epoch 154/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2004 - accuracy: 0.9172
Epoch 155/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1828 - accuracy: 0.9058
Epoch 156/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2236 - accuracy: 0.9098
Epoch 157/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2115 - accuracy: 0.9121
Epoch 158/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2044 - accuracy: 0.9087
Epoch 159/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1908 - accuracy: 0.9115
Epoch 160/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1860 - accuracy: 0.9127
Epoch 161/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1808 - accuracy: 0.9189
Epoch 162/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1896 - accuracy: 0.9155
Epoch 163/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1942 - accuracy: 0.9184
Epoch 164/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2068 - accuracy: 0.9127
Epoch 165/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2011 - accuracy: 0.9132
Epoch 166/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1994 - accuracy: 0.9115
Epoch 167/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1864 - accuracy: 0.9184
Epoch 168/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1799 - accuracy: 0.9155
Epoch 169/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1969 - accuracy: 0.9161
Epoch 170/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1879 - accuracy: 0.9161
Epoch 171/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1962 - accuracy: 0.9144
Epoch 172/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1765 - accuracy: 0.9212
Epoch 173/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1818 - accuracy: 0.9189
Epoch 174/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1977 - accuracy: 0.9167
Epoch 175/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2068 - accuracy: 0.9121
Epoch 176/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1871 - accuracy: 0.9218
Epoch 177/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1882 - accuracy: 0.9207
Epoch 178/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1952 - accuracy: 0.9172
Epoch 179/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1848 - accuracy: 0.9178
Epoch 180/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1688 - accuracy: 0.9332
Epoch 181/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1765 - accuracy: 0.9224
Epoch 182/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1958 - accuracy: 0.9172
Epoch 183/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1777 - accuracy: 0.9235
Epoch 184/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1679 - accuracy: 0.9315
Epoch 185/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1735 - accuracy: 0.9218
Epoch 186/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2022 - accuracy: 0.9241
Epoch 187/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1735 - accuracy: 0.9212
Epoch 188/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1845 - accuracy: 0.9298
Epoch 189/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1753 - accuracy: 0.9264
Epoch 190/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1813 - accuracy: 0.9252
Epoch 191/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1605 - accuracy: 0.9344
Epoch 192/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1641 - accuracy: 0.9275
Epoch 193/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1813 - accuracy: 0.9224
Epoch 194/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1761 - accuracy: 0.9207
Epoch 195/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1515 - accuracy: 0.9326
Epoch 196/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1610 - accuracy: 0.9309
Epoch 197/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1966 - accuracy: 0.9224
Epoch 198/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1788 - accuracy: 0.9258
Epoch 199/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1899 - accuracy: 0.9241
Epoch 200/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1759 - accuracy: 0.9309
Epoch 201/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1613 - accuracy: 0.9235
Epoch 202/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1634 - accuracy: 0.9304
Epoch 203/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1886 - accuracy: 0.9229
Epoch 204/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1579 - accuracy: 0.9287
Epoch 205/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1677 - accuracy: 0.9275
Epoch 206/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1632 - accuracy: 0.9344
Epoch 207/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1805 - accuracy: 0.9235
Epoch 208/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1607 - accuracy: 0.9298
Epoch 209/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1588 - accuracy: 0.9281
Epoch 210/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1558 - accuracy: 0.9389
Epoch 211/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1703 - accuracy: 0.9309
Epoch 212/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1611 - accuracy: 0.9355
Epoch 213/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1665 - accuracy: 0.9321
Epoch 214/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1639 - accuracy: 0.9241
Epoch 215/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1631 - accuracy: 0.9315
Epoch 216/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1648 - accuracy: 0.9321
Epoch 217/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1516 - accuracy: 0.9321
Epoch 218/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1774 - accuracy: 0.9252
Epoch 219/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1659 - accuracy: 0.9281
Epoch 220/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1556 - accuracy: 0.9355
Epoch 221/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1678 - accuracy: 0.9315
Epoch 222/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1436 - accuracy: 0.9418
Epoch 223/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1630 - accuracy: 0.9366
Epoch 224/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1630 - accuracy: 0.9361
Epoch 225/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1577 - accuracy: 0.9309
Epoch 226/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1553 - accuracy: 0.9326
Epoch 227/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1383 - accuracy: 0.9395
Epoch 228/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1635 - accuracy: 0.9292
Epoch 229/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1730 - accuracy: 0.9304
Epoch 230/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1640 - accuracy: 0.9292
Epoch 231/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1575 - accuracy: 0.9338
Epoch 232/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1492 - accuracy: 0.9389
Epoch 233/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1650 - accuracy: 0.9326
Epoch 234/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1427 - accuracy: 0.9424
Epoch 235/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1471 - accuracy: 0.9372
Epoch 236/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1559 - accuracy: 0.9418
Epoch 237/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1497 - accuracy: 0.9372
Epoch 238/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1687 - accuracy: 0.9315
Epoch 239/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1486 - accuracy: 0.9349
Epoch 240/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1347 - accuracy: 0.9452
Epoch 241/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1658 - accuracy: 0.9355
Epoch 242/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1522 - accuracy: 0.9412
Epoch 243/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1539 - accuracy: 0.9361
Epoch 244/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1659 - accuracy: 0.9389
Epoch 245/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1667 - accuracy: 0.9264
Epoch 246/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1448 - accuracy: 0.9418
Epoch 247/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1344 - accuracy: 0.9412
Epoch 248/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1487 - accuracy: 0.9401
Epoch 249/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1449 - accuracy: 0.9355
Epoch 250/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1416 - accuracy: 0.9332
Epoch 251/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1390 - accuracy: 0.9441
Epoch 252/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1452 - accuracy: 0.9469
Epoch 253/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1397 - accuracy: 0.9435
Epoch 254/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1419 - accuracy: 0.9452
Epoch 255/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1519 - accuracy: 0.9395
Epoch 256/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1404 - accuracy: 0.9446
Epoch 257/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1651 - accuracy: 0.9361
Epoch 258/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1413 - accuracy: 0.9429
Epoch 259/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1472 - accuracy: 0.9395
Epoch 260/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1637 - accuracy: 0.9378
Epoch 261/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1468 - accuracy: 0.9446
Epoch 262/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1296 - accuracy: 0.9475
Epoch 263/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1531 - accuracy: 0.9452
Epoch 264/300
351/351 [==============================] - 2s 4ms/step - loss: 0.1319 - accuracy: 0.9538
Epoch 265/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1576 - accuracy: 0.9481
Epoch 266/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1414 - accuracy: 0.9463
Epoch 267/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1524 - accuracy: 0.9424
Epoch 268/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1410 - accuracy: 0.9435
Epoch 269/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1371 - accuracy: 0.9458
Epoch 270/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1407 - accuracy: 0.9378
Epoch 271/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1490 - accuracy: 0.9395
Epoch 272/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1342 - accuracy: 0.9475
Epoch 273/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1419 - accuracy: 0.9481
Epoch 274/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1329 - accuracy: 0.9469
Epoch 275/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1496 - accuracy: 0.9435
Epoch 276/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1573 - accuracy: 0.9441
Epoch 277/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1435 - accuracy: 0.9418
Epoch 278/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1526 - accuracy: 0.9469
Epoch 279/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1430 - accuracy: 0.9475
Epoch 280/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1338 - accuracy: 0.9406
Epoch 281/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1288 - accuracy: 0.9469
Epoch 282/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1572 - accuracy: 0.9349
Epoch 283/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1517 - accuracy: 0.9452
Epoch 284/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1426 - accuracy: 0.9424
Epoch 285/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1507 - accuracy: 0.9361
Epoch 286/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1188 - accuracy: 0.9481
Epoch 287/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1398 - accuracy: 0.9469
Epoch 288/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1472 - accuracy: 0.9452
Epoch 289/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1653 - accuracy: 0.9418
Epoch 290/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1270 - accuracy: 0.9475
Epoch 291/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1434 - accuracy: 0.9441
Epoch 292/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1378 - accuracy: 0.9435
Epoch 293/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1246 - accuracy: 0.9526
Epoch 294/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1340 - accuracy: 0.9486
Epoch 295/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1320 - accuracy: 0.9532
Epoch 296/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1351 - accuracy: 0.9521
Epoch 297/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1385 - accuracy: 0.9503
Epoch 298/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1144 - accuracy: 0.9578
Epoch 299/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1370 - accuracy: 0.9515
Epoch 300/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1457 - accuracy: 0.9441
Epoch 1/300
351/351 [==============================] - 2s 2ms/step - loss: 0.5727 - accuracy: 0.7232
Epoch 2/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5214 - accuracy: 0.7563
Epoch 3/300
```

```
351/351 [==============================] - 1s 3ms/step - loss: 0.4952 - accuracy: 0.7637
Epoch 4/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4910 - accuracy: 0.7763
Epoch 5/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4829 - accuracy: 0.7677
Epoch 6/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4647 - accuracy: 0.7768
Epoch 7/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4746 - accuracy: 0.7785
Epoch 8/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4522 - accuracy: 0.7854
Epoch 9/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4624 - accuracy: 0.7785
Epoch 10/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4573 - accuracy: 0.7814
Epoch 11/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4436 - accuracy: 0.7791
Epoch 12/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4493 - accuracy: 0.7865
Epoch 13/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4494 - accuracy: 0.7785
Epoch 14/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4292 - accuracy: 0.7985
Epoch 15/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4302 - accuracy: 0.7900
Epoch 16/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4207 - accuracy: 0.7934
Epoch 17/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4173 - accuracy: 0.7905
Epoch 18/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4109 - accuracy: 0.7968
Epoch 19/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4047 - accuracy: 0.7979
Epoch 20/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4151 - accuracy: 0.7945
Epoch 21/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4011 - accuracy: 0.8065
Epoch 22/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3937 - accuracy: 0.7968
Epoch 23/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3949 - accuracy: 0.8014
Epoch 24/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4163 - accuracy: 0.7877
Epoch 25/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3974 - accuracy: 0.8037
Epoch 26/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3760 - accuracy: 0.8134
Epoch 27/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3866 - accuracy: 0.8111
Epoch 28/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4046 - accuracy: 0.7974
Epoch 29/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3710 - accuracy: 0.8151
Epoch 30/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3911 - accuracy: 0.8065
Epoch 31/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3768 - accuracy: 0.8116
Epoch 32/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3771 - accuracy: 0.8145
Epoch 33/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3583 - accuracy: 0.8134
Epoch 34/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3695 - accuracy: 0.8082
Epoch 35/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3539 - accuracy: 0.8168
Epoch 36/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.3557 - accuracy: 0.8162
Epoch 37/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3613 - accuracy: 0.8099
Epoch 38/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3635 - accuracy: 0.8128
Epoch 39/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3520 - accuracy: 0.8191
Epoch 40/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3494 - accuracy: 0.8134
Epoch 41/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3451 - accuracy: 0.8202
Epoch 42/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3588 - accuracy: 0.8134
Epoch 43/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3540 - accuracy: 0.8191
Epoch 44/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3455 - accuracy: 0.8236
Epoch 45/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3293 - accuracy: 0.8271
Epoch 46/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3379 - accuracy: 0.8225
Epoch 47/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3244 - accuracy: 0.8259
Epoch 48/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3226 - accuracy: 0.8299
Epoch 49/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3361 - accuracy: 0.8253
Epoch 50/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3301 - accuracy: 0.8202
Epoch 51/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3171 - accuracy: 0.8362
Epoch 52/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3148 - accuracy: 0.8345
Epoch 53/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3289 - accuracy: 0.8225
Epoch 54/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3211 - accuracy: 0.8282
Epoch 55/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3226 - accuracy: 0.8299
Epoch 56/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3310 - accuracy: 0.8293
Epoch 57/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3089 - accuracy: 0.8333
Epoch 58/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3093 - accuracy: 0.8385
Epoch 59/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3126 - accuracy: 0.8368
Epoch 60/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3034 - accuracy: 0.8328
Epoch 61/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3094 - accuracy: 0.8333
Epoch 62/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3115 - accuracy: 0.8265
Epoch 63/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3001 - accuracy: 0.8316
Epoch 64/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3068 - accuracy: 0.8373
Epoch 65/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3089 - accuracy: 0.8311
Epoch 66/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2986 - accuracy: 0.8305
Epoch 67/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2842 - accuracy: 0.8356
Epoch 68/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2982 - accuracy: 0.8350
Epoch 69/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2964 - accuracy: 0.8430
Epoch 70/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2996 - accuracy: 0.8373
Epoch 71/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2971 - accuracy: 0.8385
Epoch 72/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2800 - accuracy: 0.8447
Epoch 73/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2907 - accuracy: 0.8396
Epoch 74/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2827 - accuracy: 0.8459
Epoch 75/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2761 - accuracy: 0.8516
Epoch 76/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2856 - accuracy: 0.8584
Epoch 77/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2794 - accuracy: 0.8579
Epoch 78/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2866 - accuracy: 0.8573
Epoch 79/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2792 - accuracy: 0.8505
Epoch 80/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2728 - accuracy: 0.8527
Epoch 81/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2714 - accuracy: 0.8624
Epoch 82/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2694 - accuracy: 0.8693
Epoch 83/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2691 - accuracy: 0.8670
Epoch 84/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2766 - accuracy: 0.8602
Epoch 85/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2823 - accuracy: 0.8522
Epoch 86/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2779 - accuracy: 0.8573
Epoch 87/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2707 - accuracy: 0.8653
Epoch 88/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2728 - accuracy: 0.8505
Epoch 89/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2613 - accuracy: 0.8653
Epoch 90/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2551 - accuracy: 0.8636
Epoch 91/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2645 - accuracy: 0.8642
Epoch 92/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2619 - accuracy: 0.8642
Epoch 93/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2697 - accuracy: 0.8727
Epoch 94/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2561 - accuracy: 0.8818
Epoch 95/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2512 - accuracy: 0.8750
Epoch 96/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2442 - accuracy: 0.8796
Epoch 97/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2515 - accuracy: 0.8693
Epoch 98/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2553 - accuracy: 0.8910
Epoch 99/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2509 - accuracy: 0.8830
Epoch 100/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2491 - accuracy: 0.8647
Epoch 101/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2446 - accuracy: 0.8756
Epoch 102/300
```

```
351/351 [==============================] - 1s 3ms/step - loss: 0.2513 - accuracy: 0.8687
Epoch 103/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2762 - accuracy: 0.8716
Epoch 104/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2402 - accuracy: 0.8784
Epoch 105/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2376 - accuracy: 0.8801
Epoch 106/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2525 - accuracy: 0.8796
Epoch 107/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2395 - accuracy: 0.8836
Epoch 108/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2605 - accuracy: 0.8739
Epoch 109/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2594 - accuracy: 0.8801
Epoch 110/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2148 - accuracy: 0.8853
Epoch 111/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2434 - accuracy: 0.8739
Epoch 112/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2491 - accuracy: 0.8807
Epoch 113/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2462 - accuracy: 0.8767
Epoch 114/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2392 - accuracy: 0.8818
Epoch 115/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2421 - accuracy: 0.8779
Epoch 116/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2348 - accuracy: 0.8910
Epoch 117/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2328 - accuracy: 0.8898
Epoch 118/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2329 - accuracy: 0.8955
Epoch 119/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2379 - accuracy: 0.8893
Epoch 120/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2272 - accuracy: 0.8870
Epoch 121/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2168 - accuracy: 0.8938
Epoch 122/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2352 - accuracy: 0.8824
Epoch 123/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2193 - accuracy: 0.8921
Epoch 124/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2031 - accuracy: 0.9058
Epoch 125/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2582 - accuracy: 0.8967
Epoch 126/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2573 - accuracy: 0.8727
Epoch 127/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2468 - accuracy: 0.8898
Epoch 128/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2182 - accuracy: 0.8990
Epoch 129/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2100 - accuracy: 0.9013
Epoch 130/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2177 - accuracy: 0.8978
Epoch 131/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2243 - accuracy: 0.8967
Epoch 132/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2209 - accuracy: 0.8933
Epoch 133/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2137 - accuracy: 0.8967
Epoch 134/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2063 - accuracy: 0.8984
Epoch 135/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2357 - accuracy: 0.8933
Epoch 136/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2162 - accuracy: 0.9013
Epoch 137/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2132 - accuracy: 0.9018
Epoch 138/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2467 - accuracy: 0.8944
Epoch 139/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2165 - accuracy: 0.8898
Epoch 140/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2027 - accuracy: 0.9064
Epoch 141/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2238 - accuracy: 0.8973
Epoch 142/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2120 - accuracy: 0.9041
Epoch 143/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2036 - accuracy: 0.9058
Epoch 144/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2113 - accuracy: 0.9035
Epoch 145/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2186 - accuracy: 0.8944
Epoch 146/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2028 - accuracy: 0.9075
Epoch 147/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2079 - accuracy: 0.9053
Epoch 148/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2133 - accuracy: 0.9047
Epoch 149/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2042 - accuracy: 0.9030
Epoch 150/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2181 - accuracy: 0.9007
Epoch 151/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2056 - accuracy: 0.9047
Epoch 152/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2011 - accuracy: 0.9058
Epoch 153/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2018 - accuracy: 0.9047
Epoch 154/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2075 - accuracy: 0.9035
Epoch 155/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2062 - accuracy: 0.9104
Epoch 156/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2199 - accuracy: 0.9064
Epoch 157/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2082 - accuracy: 0.9018
Epoch 158/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1905 - accuracy: 0.9121
Epoch 159/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1918 - accuracy: 0.9150
Epoch 160/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1965 - accuracy: 0.9081
Epoch 161/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1974 - accuracy: 0.9104
Epoch 162/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1797 - accuracy: 0.9184
Epoch 163/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2195 - accuracy: 0.9041
Epoch 164/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1901 - accuracy: 0.9189
Epoch 165/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1865 - accuracy: 0.9224
Epoch 166/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2112 - accuracy: 0.9132
Epoch 167/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2228 - accuracy: 0.9035
Epoch 168/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2049 - accuracy: 0.9024
Epoch 169/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2241 - accuracy: 0.9081
Epoch 170/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2295 - accuracy: 0.9035
Epoch 171/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1934 - accuracy: 0.9104
Epoch 172/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2124 - accuracy: 0.9070
Epoch 173/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1932 - accuracy: 0.9138
Epoch 174/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1970 - accuracy: 0.9047
Epoch 175/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1929 - accuracy: 0.9092
Epoch 176/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1896 - accuracy: 0.9235
Epoch 177/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1793 - accuracy: 0.9189
Epoch 178/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1802 - accuracy: 0.9138
Epoch 179/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2050 - accuracy: 0.9144
Epoch 180/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1921 - accuracy: 0.9127
Epoch 181/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2006 - accuracy: 0.9155
Epoch 182/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2035 - accuracy: 0.9218
Epoch 183/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1831 - accuracy: 0.9201
Epoch 184/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1837 - accuracy: 0.9195
Epoch 185/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1815 - accuracy: 0.9235
Epoch 186/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1990 - accuracy: 0.9184
Epoch 187/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1788 - accuracy: 0.9218
Epoch 188/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1946 - accuracy: 0.9144
Epoch 189/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2026 - accuracy: 0.9201
Epoch 190/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1958 - accuracy: 0.9092
Epoch 191/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1772 - accuracy: 0.9252
Epoch 192/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1665 - accuracy: 0.9264
Epoch 193/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1839 - accuracy: 0.9229
Epoch 194/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1800 - accuracy: 0.9229
Epoch 195/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1736 - accuracy: 0.9224
Epoch 196/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1753 - accuracy: 0.9264
Epoch 197/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1776 - accuracy: 0.9172
Epoch 198/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1987 - accuracy: 0.9178
Epoch 199/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1892 - accuracy: 0.9178
Epoch 200/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1748 - accuracy: 0.9287
Epoch 201/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1794 - accuracy: 0.9264
Epoch 202/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1747 - accuracy: 0.9269
Epoch 203/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1847 - accuracy: 0.9207
Epoch 204/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1666 - accuracy: 0.9326
Epoch 205/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1677 - accuracy: 0.9252
Epoch 206/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1659 - accuracy: 0.9298
Epoch 207/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1915 - accuracy: 0.9201
Epoch 208/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1593 - accuracy: 0.9332
Epoch 209/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1899 - accuracy: 0.9189
Epoch 210/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1618 - accuracy: 0.9292
Epoch 211/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1721 - accuracy: 0.9229
Epoch 212/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1679 - accuracy: 0.9269
Epoch 213/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1958 - accuracy: 0.9207
Epoch 214/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1561 - accuracy: 0.9355
Epoch 215/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1593 - accuracy: 0.9321
Epoch 216/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1852 - accuracy: 0.9326
Epoch 217/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1785 - accuracy: 0.9224
Epoch 218/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1564 - accuracy: 0.9235
Epoch 219/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1698 - accuracy: 0.9321
Epoch 220/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1581 - accuracy: 0.9298
Epoch 221/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1902 - accuracy: 0.9349
Epoch 222/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1722 - accuracy: 0.9275
Epoch 223/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1784 - accuracy: 0.9258
Epoch 224/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1590 - accuracy: 0.9338
Epoch 225/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1408 - accuracy: 0.9412
Epoch 226/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1656 - accuracy: 0.9349
Epoch 227/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1518 - accuracy: 0.9361
Epoch 228/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1528 - accuracy: 0.9355
Epoch 229/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1829 - accuracy: 0.9247
Epoch 230/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1406 - accuracy: 0.9361
Epoch 231/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1640 - accuracy: 0.9372
Epoch 232/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1763 - accuracy: 0.9298
Epoch 233/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1570 - accuracy: 0.9321
Epoch 234/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1712 - accuracy: 0.9247
Epoch 235/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1497 - accuracy: 0.9361
Epoch 236/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1513 - accuracy: 0.9424
Epoch 237/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1459 - accuracy: 0.9378
Epoch 238/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1707 - accuracy: 0.9264
Epoch 239/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1482 - accuracy: 0.9349
Epoch 240/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1710 - accuracy: 0.9349
Epoch 241/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1530 - accuracy: 0.9389
Epoch 242/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1637 - accuracy: 0.9332
Epoch 243/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1536 - accuracy: 0.9361
Epoch 244/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1522 - accuracy: 0.9349
Epoch 245/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1658 - accuracy: 0.9349
Epoch 246/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1684 - accuracy: 0.9281
Epoch 247/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1558 - accuracy: 0.9366
Epoch 248/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1565 - accuracy: 0.9418
Epoch 249/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1551 - accuracy: 0.9372
Epoch 250/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1359 - accuracy: 0.9418
Epoch 251/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1706 - accuracy: 0.9292
Epoch 252/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1539 - accuracy: 0.9304
Epoch 253/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1512 - accuracy: 0.9315
Epoch 254/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1580 - accuracy: 0.9315
Epoch 255/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1586 - accuracy: 0.9338
Epoch 256/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1521 - accuracy: 0.9389
Epoch 257/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1575 - accuracy: 0.9401
Epoch 258/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1446 - accuracy: 0.9372
Epoch 259/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1504 - accuracy: 0.9338
Epoch 260/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1561 - accuracy: 0.9412
Epoch 261/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1503 - accuracy: 0.9406
Epoch 262/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1446 - accuracy: 0.9446
Epoch 263/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1498 - accuracy: 0.9412
Epoch 264/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1575 - accuracy: 0.9344
Epoch 265/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1553 - accuracy: 0.9401
Epoch 266/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1366 - accuracy: 0.9424
Epoch 267/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1475 - accuracy: 0.9418
Epoch 268/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1721 - accuracy: 0.9389
Epoch 269/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1502 - accuracy: 0.9412
Epoch 270/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1441 - accuracy: 0.9435
Epoch 271/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1327 - accuracy: 0.9435
Epoch 272/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1181 - accuracy: 0.9498
Epoch 273/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1756 - accuracy: 0.9298
Epoch 274/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1565 - accuracy: 0.9406
Epoch 275/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1570 - accuracy: 0.9429
Epoch 276/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1462 - accuracy: 0.9366
Epoch 277/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1285 - accuracy: 0.9452
Epoch 278/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1441 - accuracy: 0.9412
Epoch 279/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1446 - accuracy: 0.9406
Epoch 280/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1288 - accuracy: 0.9418
Epoch 281/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1598 - accuracy: 0.9366
Epoch 282/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1526 - accuracy: 0.9372
Epoch 283/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1372 - accuracy: 0.9475
Epoch 284/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1404 - accuracy: 0.9406
Epoch 285/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1494 - accuracy: 0.9463
Epoch 286/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1293 - accuracy: 0.9475
Epoch 287/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1549 - accuracy: 0.9401
Epoch 288/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1398 - accuracy: 0.9452
Epoch 289/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1425 - accuracy: 0.9452
Epoch 290/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1427 - accuracy: 0.9429
Epoch 291/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1389 - accuracy: 0.9435
Epoch 292/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1330 - accuracy: 0.9463
Epoch 293/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1178 - accuracy: 0.9515
Epoch 294/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1177 - accuracy: 0.9498
Epoch 295/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1305 - accuracy: 0.9469
Epoch 296/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1444 - accuracy: 0.9429
Epoch 297/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1449 - accuracy: 0.9424
Epoch 298/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1746 - accuracy: 0.9458
Epoch 299/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1437 - accuracy: 0.9521
Epoch 300/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.1150 - accuracy: 0.9521
Epoch 1/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5651 - accuracy: 0.7146
Epoch 2/300
351/351 [==============================] - 1s 2ms/step - loss: 0.5170 - accuracy: 0.7591
Epoch 3/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4962 - accuracy: 0.7643
Epoch 4/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4886 - accuracy: 0.7666
Epoch 5/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4773 - accuracy: 0.7803
Epoch 6/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4742 - accuracy: 0.7797
Epoch 7/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4618 - accuracy: 0.7860
Epoch 8/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4559 - accuracy: 0.7831
Epoch 9/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4523 - accuracy: 0.7871
Epoch 10/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4359 - accuracy: 0.7945
Epoch 11/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4404 - accuracy: 0.7917
Epoch 12/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4367 - accuracy: 0.7905
Epoch 13/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4449 - accuracy: 0.7957
Epoch 14/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4317 - accuracy: 0.7979
Epoch 15/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4256 - accuracy: 0.8037
Epoch 16/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4199 - accuracy: 0.8071
Epoch 17/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4211 - accuracy: 0.8031
Epoch 18/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4082 - accuracy: 0.8111
Epoch 19/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4186 - accuracy: 0.8037
Epoch 20/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3991 - accuracy: 0.8128
Epoch 21/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3989 - accuracy: 0.8025
Epoch 22/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3879 - accuracy: 0.8151
Epoch 23/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3872 - accuracy: 0.8134
Epoch 24/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3868 - accuracy: 0.8116
Epoch 25/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3845 - accuracy: 0.8225
Epoch 26/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3967 - accuracy: 0.8088
Epoch 27/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3893 - accuracy: 0.8179
Epoch 28/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3752 - accuracy: 0.8213
Epoch 29/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3683 - accuracy: 0.8253
Epoch 30/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3803 - accuracy: 0.8253
Epoch 31/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3744 - accuracy: 0.8174
Epoch 32/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3585 - accuracy: 0.8248
Epoch 33/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.3587 - accuracy: 0.8299
Epoch 34/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3597 - accuracy: 0.8408
Epoch 35/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3621 - accuracy: 0.8293
Epoch 36/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3461 - accuracy: 0.8339
Epoch 37/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3460 - accuracy: 0.8328
Epoch 38/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3589 - accuracy: 0.8368
Epoch 39/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3479 - accuracy: 0.8316
Epoch 40/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3481 - accuracy: 0.8316
Epoch 41/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3455 - accuracy: 0.8305
Epoch 42/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3282 - accuracy: 0.8465
Epoch 43/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3383 - accuracy: 0.8436
Epoch 44/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3288 - accuracy: 0.8493
Epoch 45/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3492 - accuracy: 0.8322
Epoch 46/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3274 - accuracy: 0.8465
Epoch 47/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3237 - accuracy: 0.8476
Epoch 48/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3162 - accuracy: 0.8487
Epoch 49/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3186 - accuracy: 0.8459
Epoch 50/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3378 - accuracy: 0.8453
Epoch 51/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3114 - accuracy: 0.8385
Epoch 52/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3121 - accuracy: 0.8527
Epoch 53/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3285 - accuracy: 0.8476
Epoch 54/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3115 - accuracy: 0.8550
Epoch 55/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3198 - accuracy: 0.8505
Epoch 56/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3317 - accuracy: 0.8390
Epoch 57/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2979 - accuracy: 0.8567
Epoch 58/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2935 - accuracy: 0.8539
Epoch 59/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3166 - accuracy: 0.8493
Epoch 60/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3159 - accuracy: 0.8539
Epoch 61/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2927 - accuracy: 0.8584
Epoch 62/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3186 - accuracy: 0.8505
Epoch 63/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2946 - accuracy: 0.8613
Epoch 64/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3003 - accuracy: 0.8516
Epoch 65/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3047 - accuracy: 0.8562
Epoch 66/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.3062 - accuracy: 0.8487
Epoch 67/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2969 - accuracy: 0.8590
Epoch 68/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2844 - accuracy: 0.8664
Epoch 69/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2941 - accuracy: 0.8602
Epoch 70/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3076 - accuracy: 0.8573
Epoch 71/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2910 - accuracy: 0.8573
Epoch 72/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2994 - accuracy: 0.8567
Epoch 73/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2994 - accuracy: 0.8522
Epoch 74/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2871 - accuracy: 0.8636
Epoch 75/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3025 - accuracy: 0.8584
Epoch 76/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2958 - accuracy: 0.8522
Epoch 77/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2870 - accuracy: 0.8573
Epoch 78/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2910 - accuracy: 0.8607
Epoch 79/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2711 - accuracy: 0.8630
Epoch 80/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2642 - accuracy: 0.8750
Epoch 81/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2977 - accuracy: 0.8619
Epoch 82/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2998 - accuracy: 0.8693
Epoch 83/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2712 - accuracy: 0.8613
Epoch 84/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2641 - accuracy: 0.8687
Epoch 85/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2654 - accuracy: 0.8739
Epoch 86/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2854 - accuracy: 0.8670
Epoch 87/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2735 - accuracy: 0.8716
Epoch 88/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2730 - accuracy: 0.8682
Epoch 89/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2541 - accuracy: 0.8750
Epoch 90/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2662 - accuracy: 0.8687
Epoch 91/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2739 - accuracy: 0.8602
Epoch 92/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2688 - accuracy: 0.8659
Epoch 93/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2735 - accuracy: 0.8659
Epoch 94/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2803 - accuracy: 0.8624
Epoch 95/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2651 - accuracy: 0.8704
Epoch 96/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2683 - accuracy: 0.8773
Epoch 97/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2686 - accuracy: 0.8699
Epoch 98/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2637 - accuracy: 0.8739
Epoch 99/300
```

```
351/351 [==============================] - 1s 2ms/step - loss: 0.2638 - accuracy: 0.8699
Epoch 100/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2526 - accuracy: 0.8739
Epoch 101/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2593 - accuracy: 0.8727
Epoch 102/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2665 - accuracy: 0.8739
Epoch 103/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2536 - accuracy: 0.8824
Epoch 104/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2736 - accuracy: 0.8773
Epoch 105/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2449 - accuracy: 0.8767
Epoch 106/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2637 - accuracy: 0.8761
Epoch 107/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2444 - accuracy: 0.8761
Epoch 108/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2656 - accuracy: 0.8779
Epoch 109/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2656 - accuracy: 0.8659
Epoch 110/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2535 - accuracy: 0.8830
Epoch 111/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2419 - accuracy: 0.8767
Epoch 112/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2517 - accuracy: 0.8744
Epoch 113/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2301 - accuracy: 0.8858
Epoch 114/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2536 - accuracy: 0.8704
Epoch 115/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2380 - accuracy: 0.8813
Epoch 116/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2468 - accuracy: 0.8773
Epoch 117/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2661 - accuracy: 0.8761
Epoch 118/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2519 - accuracy: 0.8818
Epoch 119/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2446 - accuracy: 0.8779
Epoch 120/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2377 - accuracy: 0.8836
Epoch 121/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2345 - accuracy: 0.8824
Epoch 122/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2424 - accuracy: 0.8801
Epoch 123/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2338 - accuracy: 0.8841
Epoch 124/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2306 - accuracy: 0.8813
Epoch 125/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2466 - accuracy: 0.8750
Epoch 126/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2446 - accuracy: 0.8830
Epoch 127/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2392 - accuracy: 0.8853
Epoch 128/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2300 - accuracy: 0.8813
Epoch 129/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2212 - accuracy: 0.8841
Epoch 130/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2466 - accuracy: 0.8779
Epoch 131/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2333 - accuracy: 0.8830
Epoch 132/300
```

```
351/351 [==============================] - 1s 4ms/step - loss: 0.2430 - accuracy: 0.8876
Epoch 133/300
351/351 [==============================] - 4s 11ms/step - loss: 0.2416 - accuracy: 0.883
6
Epoch 134/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2315 - accuracy: 0.8796
Epoch 135/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2039 - accuracy: 0.8904
Epoch 136/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2253 - accuracy: 0.8853
Epoch 137/300
351/351 [==============================] - 2s 6ms/step - loss: 0.2283 - accuracy: 0.8841
Epoch 138/300
351/351 [==============================] - 2s 6ms/step - loss: 0.2076 - accuracy: 0.8967
Epoch 139/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2286 - accuracy: 0.8904
Epoch 140/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2422 - accuracy: 0.8807
Epoch 141/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2230 - accuracy: 0.8830
Epoch 142/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2299 - accuracy: 0.8881
Epoch 143/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2201 - accuracy: 0.8876
Epoch 144/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2360 - accuracy: 0.8853
Epoch 145/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2163 - accuracy: 0.8881
Epoch 146/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2154 - accuracy: 0.8887
Epoch 147/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2292 - accuracy: 0.8876
Epoch 148/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2141 - accuracy: 0.8967
Epoch 149/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2357 - accuracy: 0.8853
Epoch 150/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2522 - accuracy: 0.8796
Epoch 151/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2108 - accuracy: 0.8921
Epoch 152/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2206 - accuracy: 0.8853
Epoch 153/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2418 - accuracy: 0.8818
Epoch 154/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2229 - accuracy: 0.8898
Epoch 155/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2180 - accuracy: 0.8881
Epoch 156/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2198 - accuracy: 0.8830
Epoch 157/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2018 - accuracy: 0.8916
Epoch 158/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2069 - accuracy: 0.8887
Epoch 159/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2154 - accuracy: 0.8898
Epoch 160/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2054 - accuracy: 0.8910
Epoch 161/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2045 - accuracy: 0.8933
Epoch 162/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2042 - accuracy: 0.8955
Epoch 163/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2297 - accuracy: 0.8824
Epoch 164/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2224 - accuracy: 0.8847
```

```
Epoch 165/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2087 - accuracy: 0.8961
Epoch 166/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2047 - accuracy: 0.8961
Epoch 167/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2081 - accuracy: 0.8950
Epoch 168/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2163 - accuracy: 0.8995
Epoch 169/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2056 - accuracy: 0.8933
Epoch 170/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2050 - accuracy: 0.8955
Epoch 171/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2137 - accuracy: 0.8933
Epoch 172/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2122 - accuracy: 0.8910
Epoch 173/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2163 - accuracy: 0.8927
Epoch 174/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2241 - accuracy: 0.8910
Epoch 175/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2134 - accuracy: 0.8898
Epoch 176/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2013 - accuracy: 0.8973
Epoch 177/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2013 - accuracy: 0.8973
Epoch 178/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2066 - accuracy: 0.8944
Epoch 179/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2025 - accuracy: 0.8984
Epoch 180/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2103 - accuracy: 0.8944
Epoch 181/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2104 - accuracy: 0.8944
Epoch 182/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2112 - accuracy: 0.8978
Epoch 183/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2049 - accuracy: 0.8938
Epoch 184/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2114 - accuracy: 0.8995
Epoch 185/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2033 - accuracy: 0.8893
Epoch 186/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1974 - accuracy: 0.8990
Epoch 187/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1894 - accuracy: 0.8938
Epoch 188/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2102 - accuracy: 0.8916
Epoch 189/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1860 - accuracy: 0.8984
Epoch 190/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1988 - accuracy: 0.8978
Epoch 191/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1969 - accuracy: 0.8927
Epoch 192/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1946 - accuracy: 0.9001
Epoch 193/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1996 - accuracy: 0.8933
Epoch 194/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2011 - accuracy: 0.8978
Epoch 195/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2183 - accuracy: 0.8933
Epoch 196/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1968 - accuracy: 0.9030
Epoch 197/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1906 - accuracy: 0.9041
```

```
Epoch 198/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1973 - accuracy: 0.9070
Epoch 199/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1832 - accuracy: 0.9047
Epoch 200/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1888 - accuracy: 0.9013
Epoch 201/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1881 - accuracy: 0.9035
Epoch 202/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2031 - accuracy: 0.8955
Epoch 203/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1817 - accuracy: 0.8990
Epoch 204/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1819 - accuracy: 0.9013
Epoch 205/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2185 - accuracy: 0.8990
Epoch 206/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2079 - accuracy: 0.8990
Epoch 207/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1925 - accuracy: 0.9018
Epoch 208/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1739 - accuracy: 0.9092
Epoch 209/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2109 - accuracy: 0.8950
Epoch 210/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1701 - accuracy: 0.9110
Epoch 211/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1981 - accuracy: 0.9064
Epoch 212/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1886 - accuracy: 0.8984
Epoch 213/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1798 - accuracy: 0.9024
Epoch 214/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1989 - accuracy: 0.9035
Epoch 215/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2014 - accuracy: 0.9013
Epoch 216/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1813 - accuracy: 0.9081
Epoch 217/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1916 - accuracy: 0.9053
Epoch 218/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1875 - accuracy: 0.9053
Epoch 219/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1780 - accuracy: 0.9013
Epoch 220/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1848 - accuracy: 0.9075
Epoch 221/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1809 - accuracy: 0.9001
Epoch 222/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1730 - accuracy: 0.9150
Epoch 223/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1671 - accuracy: 0.9127
Epoch 224/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1837 - accuracy: 0.9013
Epoch 225/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2008 - accuracy: 0.9001
Epoch 226/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1881 - accuracy: 0.9007
Epoch 227/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1778 - accuracy: 0.9064
Epoch 228/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1661 - accuracy: 0.9047
Epoch 229/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1927 - accuracy: 0.9047
Epoch 230/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1713 - accuracy: 0.9161
```

```
Epoch 231/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1900 - accuracy: 0.9087
Epoch 232/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1732 - accuracy: 0.9087
Epoch 233/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1850 - accuracy: 0.9092
Epoch 234/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2174 - accuracy: 0.9030
Epoch 235/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1915 - accuracy: 0.9098
Epoch 236/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1652 - accuracy: 0.9144
Epoch 237/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1915 - accuracy: 0.9058
Epoch 238/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1800 - accuracy: 0.9121
Epoch 239/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1779 - accuracy: 0.9070
Epoch 240/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1679 - accuracy: 0.9081
Epoch 241/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1731 - accuracy: 0.9058
Epoch 242/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1611 - accuracy: 0.9092
Epoch 243/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1573 - accuracy: 0.9098
Epoch 244/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1703 - accuracy: 0.9110
Epoch 245/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1563 - accuracy: 0.9172
Epoch 246/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1657 - accuracy: 0.9138
Epoch 247/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1826 - accuracy: 0.9064
Epoch 248/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1839 - accuracy: 0.9110
Epoch 249/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1616 - accuracy: 0.9121
Epoch 250/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1728 - accuracy: 0.9115
Epoch 251/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1651 - accuracy: 0.9138
Epoch 252/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1555 - accuracy: 0.9150
Epoch 253/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1622 - accuracy: 0.9092
Epoch 254/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1522 - accuracy: 0.9212
Epoch 255/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1678 - accuracy: 0.9172
Epoch 256/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1726 - accuracy: 0.9064
Epoch 257/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1748 - accuracy: 0.9161
Epoch 258/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1822 - accuracy: 0.9132
Epoch 259/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1613 - accuracy: 0.9178
Epoch 260/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1654 - accuracy: 0.9155
Epoch 261/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1677 - accuracy: 0.9138
Epoch 262/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1881 - accuracy: 0.9053
Epoch 263/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1493 - accuracy: 0.9167
```

```
Epoch 264/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1726 - accuracy: 0.9144
Epoch 265/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1351 - accuracy: 0.9247
Epoch 266/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1746 - accuracy: 0.9150
Epoch 267/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1795 - accuracy: 0.9172
Epoch 268/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1737 - accuracy: 0.9155
Epoch 269/300
351/351 [==============================] - 2s 5ms/step - loss: 0.1584 - accuracy: 0.9167
Epoch 270/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1450 - accuracy: 0.9195
Epoch 271/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1583 - accuracy: 0.9195
Epoch 272/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1759 - accuracy: 0.9212
Epoch 273/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1537 - accuracy: 0.9189
Epoch 274/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1510 - accuracy: 0.9212
Epoch 275/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1628 - accuracy: 0.9172
Epoch 276/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1733 - accuracy: 0.9172
Epoch 277/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1488 - accuracy: 0.9292
Epoch 278/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1587 - accuracy: 0.9252
Epoch 279/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1618 - accuracy: 0.9178
Epoch 280/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1486 - accuracy: 0.9229
Epoch 281/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1519 - accuracy: 0.9207
Epoch 282/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1441 - accuracy: 0.9269
Epoch 283/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1471 - accuracy: 0.9247
Epoch 284/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1583 - accuracy: 0.9178
Epoch 285/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1448 - accuracy: 0.9212
Epoch 286/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1574 - accuracy: 0.9195
Epoch 287/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1672 - accuracy: 0.9167
Epoch 288/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1508 - accuracy: 0.9184
Epoch 289/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1493 - accuracy: 0.9241
Epoch 290/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1523 - accuracy: 0.9264
Epoch 291/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1699 - accuracy: 0.9184
Epoch 292/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1513 - accuracy: 0.9229
Epoch 293/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1411 - accuracy: 0.9241
Epoch 294/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1652 - accuracy: 0.9258
Epoch 295/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1704 - accuracy: 0.9229
Epoch 296/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1450 - accuracy: 0.9275
```

```
Epoch 297/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1626 - accuracy: 0.9212
Epoch 298/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1361 - accuracy: 0.9258
Epoch 299/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1456 - accuracy: 0.9269
Epoch 300/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1488 - accuracy: 0.9247
Epoch 1/300
351/351 [==============================] - 2s 3ms/step - loss: 0.5714 - accuracy: 0.7026
Epoch 2/300
351/351 [==============================] - 1s 3ms/step - loss: 0.5222 - accuracy: 0.7420
Epoch 3/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4958 - accuracy: 0.7614
Epoch 4/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4754 - accuracy: 0.7671
Epoch 5/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4811 - accuracy: 0.7683
Epoch 6/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4654 - accuracy: 0.7711
Epoch 7/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4662 - accuracy: 0.7780
Epoch 8/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4690 - accuracy: 0.7723
Epoch 9/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4541 - accuracy: 0.7831
Epoch 10/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4584 - accuracy: 0.7740
Epoch 11/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4353 - accuracy: 0.7894
Epoch 12/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4508 - accuracy: 0.7803
Epoch 13/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4385 - accuracy: 0.7911
Epoch 14/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4238 - accuracy: 0.7974
Epoch 15/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4344 - accuracy: 0.7831
Epoch 16/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4287 - accuracy: 0.7888
Epoch 17/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4174 - accuracy: 0.7997
Epoch 18/300
351/351 [==============================] - 1s 2ms/step - loss: 0.4352 - accuracy: 0.7939
Epoch 19/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4099 - accuracy: 0.8059
Epoch 20/300
351/351 [==============================] - 1s 4ms/step - loss: 0.4067 - accuracy: 0.7968
Epoch 21/300
351/351 [==============================] - 1s 4ms/step - loss: 0.4095 - accuracy: 0.7957
Epoch 22/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4031 - accuracy: 0.8071
Epoch 23/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3936 - accuracy: 0.8008
Epoch 24/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4072 - accuracy: 0.7962
Epoch 25/300
351/351 [==============================] - 1s 3ms/step - loss: 0.4017 - accuracy: 0.7968
Epoch 26/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3917 - accuracy: 0.8082
Epoch 27/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3874 - accuracy: 0.8105
Epoch 28/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3763 - accuracy: 0.8105
Epoch 29/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3894 - accuracy: 0.8088
```

```
Epoch 30/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3916 - accuracy: 0.8105
Epoch 31/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3833 - accuracy: 0.8191
Epoch 32/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3760 - accuracy: 0.8191
Epoch 33/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3698 - accuracy: 0.8134
Epoch 34/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3759 - accuracy: 0.8168
Epoch 35/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3735 - accuracy: 0.8162
Epoch 36/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3692 - accuracy: 0.8179
Epoch 37/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3600 - accuracy: 0.8208
Epoch 38/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3610 - accuracy: 0.8276
Epoch 39/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3654 - accuracy: 0.8213
Epoch 40/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3635 - accuracy: 0.8242
Epoch 41/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3658 - accuracy: 0.8128
Epoch 42/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3546 - accuracy: 0.8208
Epoch 43/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3642 - accuracy: 0.8299
Epoch 44/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3439 - accuracy: 0.8196
Epoch 45/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3635 - accuracy: 0.8248
Epoch 46/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3448 - accuracy: 0.8316
Epoch 47/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3495 - accuracy: 0.8293
Epoch 48/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3571 - accuracy: 0.8253
Epoch 49/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3512 - accuracy: 0.8282
Epoch 50/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3272 - accuracy: 0.8299
Epoch 51/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3438 - accuracy: 0.8253
Epoch 52/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3409 - accuracy: 0.8299
Epoch 53/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3244 - accuracy: 0.8316
Epoch 54/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3339 - accuracy: 0.8419
Epoch 55/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3415 - accuracy: 0.8276
Epoch 56/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3390 - accuracy: 0.8311
Epoch 57/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3235 - accuracy: 0.8311
Epoch 58/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3207 - accuracy: 0.8362
Epoch 59/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3231 - accuracy: 0.8379
Epoch 60/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3245 - accuracy: 0.8311
Epoch 61/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3247 - accuracy: 0.8390
Epoch 62/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3173 - accuracy: 0.8316
```

```
Epoch 63/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3274 - accuracy: 0.8396
Epoch 64/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3165 - accuracy: 0.8390
Epoch 65/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3197 - accuracy: 0.8459
Epoch 66/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3070 - accuracy: 0.8350
Epoch 67/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3198 - accuracy: 0.8305
Epoch 68/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3073 - accuracy: 0.8430
Epoch 69/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2977 - accuracy: 0.8413
Epoch 70/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2993 - accuracy: 0.8499
Epoch 71/300
351/351 [==============================] - 1s 2ms/step - loss: 0.3083 - accuracy: 0.8522
Epoch 72/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2937 - accuracy: 0.8522
Epoch 73/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2991 - accuracy: 0.8539
Epoch 74/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2986 - accuracy: 0.8493
Epoch 75/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2844 - accuracy: 0.8539
Epoch 76/300
351/351 [==============================] - 1s 3ms/step - loss: 0.3052 - accuracy: 0.8436
Epoch 77/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2903 - accuracy: 0.8476
Epoch 78/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2903 - accuracy: 0.8487
Epoch 79/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2879 - accuracy: 0.8510
Epoch 80/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2872 - accuracy: 0.8482
Epoch 81/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2925 - accuracy: 0.8533
Epoch 82/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2868 - accuracy: 0.8545
Epoch 83/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2605 - accuracy: 0.8607
Epoch 84/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2898 - accuracy: 0.8505
Epoch 85/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2833 - accuracy: 0.8590
Epoch 86/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2719 - accuracy: 0.8550
Epoch 87/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2719 - accuracy: 0.8584
Epoch 88/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2735 - accuracy: 0.8562
Epoch 89/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2720 - accuracy: 0.8545
Epoch 90/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2671 - accuracy: 0.8556
Epoch 91/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2636 - accuracy: 0.8624
Epoch 92/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2655 - accuracy: 0.8533
Epoch 93/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2697 - accuracy: 0.8556
Epoch 94/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2722 - accuracy: 0.8527
Epoch 95/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2782 - accuracy: 0.8567
```

```
Epoch 96/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2835 - accuracy: 0.8584
Epoch 97/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2657 - accuracy: 0.8613
Epoch 98/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2593 - accuracy: 0.8584
Epoch 99/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2695 - accuracy: 0.8579
Epoch 100/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2569 - accuracy: 0.8647
Epoch 101/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2676 - accuracy: 0.8567
Epoch 102/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2756 - accuracy: 0.8556
Epoch 103/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2588 - accuracy: 0.8602
Epoch 104/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2497 - accuracy: 0.8647
Epoch 105/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2530 - accuracy: 0.8659
Epoch 106/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2537 - accuracy: 0.8590
Epoch 107/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2680 - accuracy: 0.8613
Epoch 108/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2626 - accuracy: 0.8630
Epoch 109/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2604 - accuracy: 0.8590
Epoch 110/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2680 - accuracy: 0.8642
Epoch 111/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2482 - accuracy: 0.8636
Epoch 112/300
351/351 [==============================] - 1s 4ms/step - loss: 0.2597 - accuracy: 0.8573
Epoch 113/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2578 - accuracy: 0.8659
Epoch 114/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2418 - accuracy: 0.8607
Epoch 115/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2430 - accuracy: 0.8670
Epoch 116/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2610 - accuracy: 0.8624
Epoch 117/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2447 - accuracy: 0.8670
Epoch 118/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2325 - accuracy: 0.8733
Epoch 119/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2690 - accuracy: 0.8562
Epoch 120/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2584 - accuracy: 0.8590
Epoch 121/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2499 - accuracy: 0.8642
Epoch 122/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2333 - accuracy: 0.8727
Epoch 123/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2413 - accuracy: 0.8727
Epoch 124/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2475 - accuracy: 0.8659
Epoch 125/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2419 - accuracy: 0.8699
Epoch 126/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2341 - accuracy: 0.8693
Epoch 127/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2349 - accuracy: 0.8676
Epoch 128/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2515 - accuracy: 0.8613
```

```
Epoch 129/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2327 - accuracy: 0.8630
Epoch 130/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2445 - accuracy: 0.8670
Epoch 131/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2350 - accuracy: 0.8653
Epoch 132/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2375 - accuracy: 0.8693
Epoch 133/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2419 - accuracy: 0.8710
Epoch 134/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2236 - accuracy: 0.8642
Epoch 135/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2381 - accuracy: 0.8682
Epoch 136/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2382 - accuracy: 0.8659
Epoch 137/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2212 - accuracy: 0.8710
Epoch 138/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2527 - accuracy: 0.8682
Epoch 139/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2254 - accuracy: 0.8733
Epoch 140/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2252 - accuracy: 0.8756
Epoch 141/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2397 - accuracy: 0.8721
Epoch 142/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2456 - accuracy: 0.8710
Epoch 143/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2309 - accuracy: 0.8721
Epoch 144/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2359 - accuracy: 0.8664
Epoch 145/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2345 - accuracy: 0.8744
Epoch 146/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2252 - accuracy: 0.8693
Epoch 147/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2295 - accuracy: 0.8664
Epoch 148/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2296 - accuracy: 0.8721
Epoch 149/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2240 - accuracy: 0.8847
Epoch 150/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2256 - accuracy: 0.8881
Epoch 151/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2266 - accuracy: 0.8784
Epoch 152/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2246 - accuracy: 0.8773
Epoch 153/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2063 - accuracy: 0.8904
Epoch 154/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2125 - accuracy: 0.8921
Epoch 155/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2239 - accuracy: 0.8910
Epoch 156/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2268 - accuracy: 0.8921
Epoch 157/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2060 - accuracy: 0.8921
Epoch 158/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2388 - accuracy: 0.8796
Epoch 159/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2226 - accuracy: 0.8944
Epoch 160/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2048 - accuracy: 0.8904
Epoch 161/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2180 - accuracy: 0.8927
```

```
Epoch 162/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2072 - accuracy: 0.9007
Epoch 163/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2142 - accuracy: 0.8876
Epoch 164/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2134 - accuracy: 0.9001
Epoch 165/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2144 - accuracy: 0.8961
Epoch 166/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2103 - accuracy: 0.8944
Epoch 167/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2058 - accuracy: 0.8904
Epoch 168/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2148 - accuracy: 0.8841
Epoch 169/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2170 - accuracy: 0.8887
Epoch 170/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2083 - accuracy: 0.8984
Epoch 171/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2073 - accuracy: 0.8955
Epoch 172/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2269 - accuracy: 0.8818
Epoch 173/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1966 - accuracy: 0.8961
Epoch 174/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2107 - accuracy: 0.8887
Epoch 175/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2092 - accuracy: 0.8933
Epoch 176/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2182 - accuracy: 0.8990
Epoch 177/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2282 - accuracy: 0.8933
Epoch 178/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2127 - accuracy: 0.8813
Epoch 179/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2119 - accuracy: 0.9007
Epoch 180/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2013 - accuracy: 0.8978
Epoch 181/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2177 - accuracy: 0.8978
Epoch 182/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1909 - accuracy: 0.9110
Epoch 183/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1902 - accuracy: 0.9013
Epoch 184/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2100 - accuracy: 0.8990
Epoch 185/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2024 - accuracy: 0.9047
Epoch 186/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2003 - accuracy: 0.9024
Epoch 187/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2119 - accuracy: 0.9024
Epoch 188/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1897 - accuracy: 0.9007
Epoch 189/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1880 - accuracy: 0.9041
Epoch 190/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2109 - accuracy: 0.8950
Epoch 191/300
351/351 [==============================] - 1s 3ms/step - loss: 0.2047 - accuracy: 0.8950
Epoch 192/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1960 - accuracy: 0.9030
Epoch 193/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2117 - accuracy: 0.9001
Epoch 194/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2083 - accuracy: 0.8921
```

```
Epoch 195/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2082 - accuracy: 0.9035
Epoch 196/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1862 - accuracy: 0.9058
Epoch 197/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1918 - accuracy: 0.9058
Epoch 198/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1959 - accuracy: 0.9064
Epoch 199/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1878 - accuracy: 0.9184
Epoch 200/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2033 - accuracy: 0.9115
Epoch 201/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2012 - accuracy: 0.9127
Epoch 202/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1899 - accuracy: 0.9070
Epoch 203/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1802 - accuracy: 0.9070
Epoch 204/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1951 - accuracy: 0.9024
Epoch 205/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2068 - accuracy: 0.9070
Epoch 206/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1851 - accuracy: 0.9121
Epoch 207/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1875 - accuracy: 0.9115
Epoch 208/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1918 - accuracy: 0.9070
Epoch 209/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1828 - accuracy: 0.9150
Epoch 210/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1837 - accuracy: 0.9189
Epoch 211/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1770 - accuracy: 0.9212
Epoch 212/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1934 - accuracy: 0.9195
Epoch 213/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1681 - accuracy: 0.9218
Epoch 214/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1968 - accuracy: 0.9161
Epoch 215/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1751 - accuracy: 0.9189
Epoch 216/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1873 - accuracy: 0.9127
Epoch 217/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1662 - accuracy: 0.9218
Epoch 218/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1981 - accuracy: 0.9144
Epoch 219/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1758 - accuracy: 0.9212
Epoch 220/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1961 - accuracy: 0.9161
Epoch 221/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1853 - accuracy: 0.9184
Epoch 222/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1898 - accuracy: 0.9189
Epoch 223/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1916 - accuracy: 0.9247
Epoch 224/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1802 - accuracy: 0.9264
Epoch 225/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1712 - accuracy: 0.9189
Epoch 226/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1983 - accuracy: 0.9081
Epoch 227/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1756 - accuracy: 0.9201
```

```
Epoch 228/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1769 - accuracy: 0.9189
Epoch 229/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1696 - accuracy: 0.9258
Epoch 230/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1720 - accuracy: 0.9132
Epoch 231/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1945 - accuracy: 0.9201
Epoch 232/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1919 - accuracy: 0.9115
Epoch 233/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1941 - accuracy: 0.9201
Epoch 234/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1759 - accuracy: 0.9218
Epoch 235/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1693 - accuracy: 0.9247
Epoch 236/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1953 - accuracy: 0.9138
Epoch 237/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2029 - accuracy: 0.9098
Epoch 238/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1697 - accuracy: 0.9264
Epoch 239/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1689 - accuracy: 0.9264
Epoch 240/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1725 - accuracy: 0.9218
Epoch 241/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1706 - accuracy: 0.9212
Epoch 242/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1604 - accuracy: 0.9315
Epoch 243/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1690 - accuracy: 0.9252
Epoch 244/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1830 - accuracy: 0.9224
Epoch 245/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1602 - accuracy: 0.9218
Epoch 246/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1620 - accuracy: 0.9229
Epoch 247/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1800 - accuracy: 0.9252
Epoch 248/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1713 - accuracy: 0.9252
Epoch 249/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1717 - accuracy: 0.9189
Epoch 250/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1782 - accuracy: 0.9207
Epoch 251/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1600 - accuracy: 0.9321
Epoch 252/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1860 - accuracy: 0.9201
Epoch 253/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1781 - accuracy: 0.9224
Epoch 254/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1850 - accuracy: 0.9195
Epoch 255/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1798 - accuracy: 0.9172
Epoch 256/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1738 - accuracy: 0.9229
Epoch 257/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1781 - accuracy: 0.9224
Epoch 258/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1582 - accuracy: 0.9264
Epoch 259/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1494 - accuracy: 0.9401
Epoch 260/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1773 - accuracy: 0.9207
```

```
Epoch 261/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1551 - accuracy: 0.9304
Epoch 262/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1615 - accuracy: 0.9321
Epoch 263/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1638 - accuracy: 0.9258
Epoch 264/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1678 - accuracy: 0.9229
Epoch 265/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1518 - accuracy: 0.9292
Epoch 266/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1735 - accuracy: 0.9241
Epoch 267/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1639 - accuracy: 0.9321
Epoch 268/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1671 - accuracy: 0.9292
Epoch 269/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1522 - accuracy: 0.9366
Epoch 270/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1814 - accuracy: 0.9189
Epoch 271/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1953 - accuracy: 0.9184
Epoch 272/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1614 - accuracy: 0.9332
Epoch 273/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1699 - accuracy: 0.9247
Epoch 274/300
351/351 [==============================] - 1s 2ms/step - loss: 0.2023 - accuracy: 0.9184
Epoch 275/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1528 - accuracy: 0.9292
Epoch 276/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1628 - accuracy: 0.9366
Epoch 277/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1656 - accuracy: 0.9424
Epoch 278/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1694 - accuracy: 0.9287
Epoch 279/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1571 - accuracy: 0.9326
Epoch 280/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1696 - accuracy: 0.9247
Epoch 281/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1699 - accuracy: 0.9321
Epoch 282/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1555 - accuracy: 0.9275
Epoch 283/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1788 - accuracy: 0.9252
Epoch 284/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1679 - accuracy: 0.9269
Epoch 285/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1925 - accuracy: 0.9269
Epoch 286/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1488 - accuracy: 0.9281
Epoch 287/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1656 - accuracy: 0.9298
Epoch 288/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1535 - accuracy: 0.9309
Epoch 289/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1405 - accuracy: 0.9401
Epoch 290/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1890 - accuracy: 0.9247
Epoch 291/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1533 - accuracy: 0.9321
Epoch 292/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1492 - accuracy: 0.9326
Epoch 293/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1509 - accuracy: 0.9332
```

```
Epoch 294/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1516 - accuracy: 0.9429
Epoch 295/300
351/351 [==============================] - 1s 3ms/step - loss: 0.1548 - accuracy: 0.9349
Epoch 296/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1552 - accuracy: 0.9292
Epoch 297/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1487 - accuracy: 0.9332
Epoch 298/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1777 - accuracy: 0.9338
Epoch 299/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1719 - accuracy: 0.9269
Epoch 300/300
351/351 [==============================] - 1s 2ms/step - loss: 0.1530 - accuracy: 0.9441
Cross-validation mean accuracy: 0.7264840182648402, std: 0.014611872146118704
```

In [64]:
```python
from sklearn.model_selection import cross_val_score

# Perform 5-fold cross validation
scores = cross_val_score(pipe, X_resampled, y_resampled, cv=5)

# Print the mean cross-validation score
print("Cross-Validation Mean Score:", scores.mean())
```

```
Cross-Validation Mean Score: 0.7680365296803653
```

Model That Performs Best (as of December 1st, 2023)

Model stacking with the following base models perform best:

- **Linear SVM** W/ C=10
- **Adaptive Boosting** (AdaBoost) W/ default parameter values
- **K-Nearest Neighbors** W/ K=1 and Manhattan Distance (L1 Norm)
- **Random Forest** W/ 10 Trees
- (Extreme) **Gradient Boosting** W/ A maximum tree depth of 7, learning rate of 0.2, and subsample of 0.5

and the final estimator was a **Linear SVM** with the same parameters as before.

For computational efficiency, RandomizedSearchCV was used as opposed to GridSearchCV, however a lot of hyperparameter tuning came from trial and error as well.

Accuracies

We get a model accuracy of **~73.164%** and a balanced accuracy of **~78.059%**

After performing **5-fold** cross-validation, we get a mean CV score of **~77.2603%**

Results:

Kaggle *Public* Score -> **75.228%**\ Kaggle *Private* Score -> **79.611%** -> **4th Highest Score!**

In [12]:
```python
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import StackingClassifier, AdaBoostClassifier, GradientBoostingCla
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from imblearn.over_sampling import ADASYN
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
import time

# define base models
base_models = [
    ("svc", SVC(kernel='linear', C=10.0, probability=True, random_state=2)),
    ("ada", AdaBoostClassifier(n_estimators=50, learning_rate=1.0, random_state=2)),
    ("knn", KNeighborsClassifier(n_neighbors=1, weights='uniform', algorithm='auto', lea
    ("rf", RandomForestClassifier(random_state=1, n_estimators=10, bootstrap=True)),
    ("xgb", XGBClassifier(random_state=2, max_depth=7, learning_rate=0.2, n_estimators=1
]

# Start timer
start_time = time.time()

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()),
     ("model", StackingClassifier(estimators=base_models, final_estimator=SVC(kernel='li
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "fw_distance", "compartment_synapse_prop", "axon_dendri
    "compartment_apical",
    "compartment_apical_shaft",
    "compartment_apical_tuft",
    "compartment_axon",
    "compartment_basal",
    "compartment_oblique",
    "compartment_soma",
    "pre_apical_count",
    "pre_apical_shaft_count",
    "pre_apical_tuft_count",
    "pre_axon_count",
    "pre_basal_count",
    "pre_oblique_count",
    "pre_soma_count",
    "post_apical_count",
    "post_apical_shaft_count",
    "post_apical_tuft_count",
    "post_axon_count",
    "post_basal_count",
    "post_oblique_count",
    "post_soma_count"]],
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "fw_distance", "compa
    "compartment_apical",
    "compartment_apical_shaft",
    "compartment_apical_tuft",
    "compartment_axon",
    "compartment_basal",
    "compartment_oblique",
    "compartment_soma",
    "pre_apical_count",
    "pre_apical_shaft_count",
    "pre_apical_tuft_count",
    "pre_axon_count",
    "pre_basal_count",
    "pre_oblique_count",
    "pre_soma_count",
    "post_apical_count",
    "post_apical_shaft_count",
    "post_apical_tuft_count",
    "post_axon_count",
    "post_basal_count",
    "post_oblique_count",
    "post_soma_count"]])[:, 1]

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)

# Stop timer and print time
end_time = time.time()
print(f"Time taken: {end_time - start_time} seconds")
```

```
accuracy: 0.7316436623886782
[[26968  9928]
 [   46   225]]
balanced accuracy: 0.7805888217164912
Time taken: 119.38371515274048 seconds
```

In [23]:
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, randint

# define the parameter distribution
param_dist = {
    'model__svc__C': uniform(0.1, 10),
    'model__ada__n_estimators': randint(30, 70),
    'model__ada__learning_rate': uniform(0.5, 1.5),
    'model__knn__n_neighbors': randint(1, 5),
    'model__rf__n_estimators': randint(10, 100),
    'model__rf__max_depth': [None, 5, 10],
    'model__xgb__max_depth': randint(5, 9),
    'model__xgb__learning_rate': uniform(0.1, 0.3),
    'model__xgb__n_estimators': randint(50, 150),
    'model__xgb__subsample': uniform(0.3, 0.7)
}

# create a RandomizedSearchCV object
random_search = RandomizedSearchCV(estimator=pipe, param_distributions=param_dist, n_ite

# fit the model
random_search.fit(X_resampled, y_resampled)

# print the best parameters
best_params = random_search.best_params_
for param, value in best_params.items():
    print(f"{param}: {value}")
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END model__ada__learning_rate=1.1539923532130056, model__ada__n_estimators=38, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=85, model__svc__
C=4.30367802087489, model__xgb__learning_rate=0.19910044630116225, model__xgb__max_depth
=8, model__xgb__n_estimators=125, model__xgb__subsample=0.7334896764454646; total time=
52.4s
[CV] END model__ada__learning_rate=1.1539923532130056, model__ada__n_estimators=38, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=85, model__svc__
C=4.30367802087489, model__xgb__learning_rate=0.19910044630116225, model__xgb__max_depth
=8, model__xgb__n_estimators=125, model__xgb__subsample=0.7334896764454646; total time=
54.3s
[CV] END model__ada__learning_rate=1.1539923532130056, model__ada__n_estimators=38, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=85, model__svc__
C=4.30367802087489, model__xgb__learning_rate=0.19910044630116225, model__xgb__max_depth
=8, model__xgb__n_estimators=125, model__xgb__subsample=0.7334896764454646; total time=
49.9s
[CV] END model__ada__learning_rate=1.1539923532130056, model__ada__n_estimators=38, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=85, model__svc__
C=4.30367802087489, model__xgb__learning_rate=0.19910044630116225, model__xgb__max_depth
=8, model__xgb__n_estimators=125, model__xgb__subsample=0.7334896764454646; total time=
53.3s
[CV] END model__ada__learning_rate=1.1539923532130056, model__ada__n_estimators=38, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=85, model__svc__
C=4.30367802087489, model__xgb__learning_rate=0.19910044630116225, model__xgb__max_depth
=8, model__xgb__n_estimators=125, model__xgb__subsample=0.7334896764454646; total time=
51.6s
[CV] END model__ada__learning_rate=0.9494820105117847, model__ada__n_estimators=56, mode
l__knn__n_neighbors=1, model__rf__max_depth=None, model__rf__n_estimators=47, model__svc
__C=6.930259944094713, model__xgb__learning_rate=0.2495683510159957, model__xgb__max_dep
th=7, model__xgb__n_estimators=101, model__xgb__subsample=0.8497346034716713; total time
= 1.2min
[CV] END model__ada__learning_rate=0.9494820105117847, model__ada__n_estimators=56, mode
l__knn__n_neighbors=1, model__rf__max_depth=None, model__rf__n_estimators=47, model__svc
__C=6.930259944094713, model__xgb__learning_rate=0.2495683510159957, model__xgb__max_dep
th=7, model__xgb__n_estimators=101, model__xgb__subsample=0.8497346034716713; total time
= 1.2min
[CV] END model__ada__learning_rate=0.9494820105117847, model__ada__n_estimators=56, mode
l__knn__n_neighbors=1, model__rf__max_depth=None, model__rf__n_estimators=47, model__svc
__C=6.930259944094713, model__xgb__learning_rate=0.2495683510159957, model__xgb__max_dep
th=7, model__xgb__n_estimators=101, model__xgb__subsample=0.8497346034716713; total time
= 1.1min
[CV] END model__ada__learning_rate=0.9494820105117847, model__ada__n_estimators=56, mode
l__knn__n_neighbors=1, model__rf__max_depth=None, model__rf__n_estimators=47, model__svc
__C=6.930259944094713, model__xgb__learning_rate=0.2495683510159957, model__xgb__max_dep
th=7, model__xgb__n_estimators=101, model__xgb__subsample=0.8497346034716713; total time
= 1.2min
[CV] END model__ada__learning_rate=0.9494820105117847, model__ada__n_estimators=56, mode
l__knn__n_neighbors=1, model__rf__max_depth=None, model__rf__n_estimators=47, model__svc
__C=6.930259944094713, model__xgb__learning_rate=0.2495683510159957, model__xgb__max_dep
th=7, model__xgb__n_estimators=101, model__xgb__subsample=0.8497346034716713; total time
= 1.3min
[CV] END model__ada__learning_rate=1.7809629389592332, model__ada__n_estimators=33, mode
l__knn__n_neighbors=2, model__rf__max_depth=None, model__rf__n_estimators=78, model__svc
__C=3.070183571076878, model__xgb__learning_rate=0.1863606447610013, model__xgb__max_dep
th=6, model__xgb__n_estimators=81, model__xgb__subsample=0.42720892644682507; total time
=  45.0s
[CV] END model__ada__learning_rate=1.7809629389592332, model__ada__n_estimators=33, mode
l__knn__n_neighbors=2, model__rf__max_depth=None, model__rf__n_estimators=78, model__svc
__C=3.070183571076878, model__xgb__learning_rate=0.1863606447610013, model__xgb__max_dep
th=6, model__xgb__n_estimators=81, model__xgb__subsample=0.42720892644682507; total time
=  50.2s
[CV] END model__ada__learning_rate=1.7809629389592332, model__ada__n_estimators=33, mode
l__knn__n_neighbors=2, model__rf__max_depth=None, model__rf__n_estimators=78, model__svc
__C=3.070183571076878, model__xgb__learning_rate=0.1863606447610013, model__xgb__max_dep
th=6, model__xgb__n_estimators=81, model__xgb__subsample=0.42720892644682507; total time
=  43.7s
```

```
[CV] END model__ada__learning_rate=1.7809629389592332, model__ada__n_estimators=33, mode
l__knn__n_neighbors=2, model__rf__max_depth=None, model__rf__n_estimators=78, model__svc
__C=3.070183571076878, model__xgb__learning_rate=0.1863606447610013, model__xgb__max_dep
th=6, model__xgb__n_estimators=81, model__xgb__subsample=0.42720892644682507; total time
=  42.3s
[CV] END model__ada__learning_rate=1.7809629389592332, model__ada__n_estimators=33, mode
l__knn__n_neighbors=2, model__rf__max_depth=None, model__rf__n_estimators=78, model__svc
__C=3.070183571076878, model__xgb__learning_rate=0.1863606447610013, model__xgb__max_dep
th=6, model__xgb__n_estimators=81, model__xgb__subsample=0.42720892644682507; total time
=  47.6s
[CV] END model__ada__learning_rate=1.2414346515768613, model__ada__n_estimators=42, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=73, model__svc
__C=2.303062070705597, model__xgb__learning_rate=0.2049478855009898, model__xgb__max_dep
th=7, model__xgb__n_estimators=58, model__xgb__subsample=0.44122025838547574; total time
=  42.1s
[CV] END model__ada__learning_rate=1.2414346515768613, model__ada__n_estimators=42, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=73, model__svc
__C=2.303062070705597, model__xgb__learning_rate=0.2049478855009898, model__xgb__max_dep
th=7, model__xgb__n_estimators=58, model__xgb__subsample=0.44122025838547574; total time
=  39.8s
[CV] END model__ada__learning_rate=1.2414346515768613, model__ada__n_estimators=42, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=73, model__svc
__C=2.303062070705597, model__xgb__learning_rate=0.2049478855009898, model__xgb__max_dep
th=7, model__xgb__n_estimators=58, model__xgb__subsample=0.44122025838547574; total time
=  38.5s
[CV] END model__ada__learning_rate=1.2414346515768613, model__ada__n_estimators=42, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=73, model__svc
__C=2.303062070705597, model__xgb__learning_rate=0.2049478855009898, model__xgb__max_dep
th=7, model__xgb__n_estimators=58, model__xgb__subsample=0.44122025838547574; total time
=  37.9s
[CV] END model__ada__learning_rate=1.2414346515768613, model__ada__n_estimators=42, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=73, model__svc
__C=2.303062070705597, model__xgb__learning_rate=0.2049478855009898, model__xgb__max_dep
th=7, model__xgb__n_estimators=58, model__xgb__subsample=0.44122025838547574; total time
=  37.3s
[CV] END model__ada__learning_rate=1.4606100878223722, model__ada__n_estimators=38, mode
l__knn__n_neighbors=4, model__rf__max_depth=5, model__rf__n_estimators=32, model__svc__C
=8.036374544415771, model__xgb__learning_rate=0.274001253663342, model__xgb__max_depth=
6, model__xgb__n_estimators=140, model__xgb__subsample=0.6566681135446268; total time=
1.2min
[CV] END model__ada__learning_rate=1.4606100878223722, model__ada__n_estimators=38, mode
l__knn__n_neighbors=4, model__rf__max_depth=5, model__rf__n_estimators=32, model__svc__C
=8.036374544415771, model__xgb__learning_rate=0.274001253663342, model__xgb__max_depth=
6, model__xgb__n_estimators=140, model__xgb__subsample=0.6566681135446268; total time=
1.3min
[CV] END model__ada__learning_rate=1.4606100878223722, model__ada__n_estimators=38, mode
l__knn__n_neighbors=4, model__rf__max_depth=5, model__rf__n_estimators=32, model__svc__C
=8.036374544415771, model__xgb__learning_rate=0.274001253663342, model__xgb__max_depth=
6, model__xgb__n_estimators=140, model__xgb__subsample=0.6566681135446268; total time=
1.2min
[CV] END model__ada__learning_rate=1.4606100878223722, model__ada__n_estimators=38, mode
l__knn__n_neighbors=4, model__rf__max_depth=5, model__rf__n_estimators=32, model__svc__C
=8.036374544415771, model__xgb__learning_rate=0.274001253663342, model__xgb__max_depth=
6, model__xgb__n_estimators=140, model__xgb__subsample=0.6566681135446268; total time=
1.2min
[CV] END model__ada__learning_rate=1.4606100878223722, model__ada__n_estimators=38, mode
l__knn__n_neighbors=4, model__rf__max_depth=5, model__rf__n_estimators=32, model__svc__C
=8.036374544415771, model__xgb__learning_rate=0.274001253663342, model__xgb__max_depth=
6, model__xgb__n_estimators=140, model__xgb__subsample=0.6566681135446268; total time=
1.2min
[CV] END model__ada__learning_rate=0.8259048896601802, model__ada__n_estimators=62, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=18, model__svc__
C=2.6873317142982747, model__xgb__learning_rate=0.34275258579123147, model__xgb__max_dep
th=7, model__xgb__n_estimators=110, model__xgb__subsample=0.627489745430074; total time=
  38.0s
[CV] END model__ada__learning_rate=0.8259048896601802, model__ada__n_estimators=62, mode
```

```
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=18, model__svc__
C=2.6873317142982747, model__xgb__learning_rate=0.34275258579123147, model__xgb__max_dep
th=7, model__xgb__n_estimators=110, model__xgb__subsample=0.627489745430074; total time=
  37.5s
[CV] END model__ada__learning_rate=0.8259048896601802, model__ada__n_estimators=62, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=18, model__svc__
C=2.6873317142982747, model__xgb__learning_rate=0.34275258579123147, model__xgb__max_dep
th=7, model__xgb__n_estimators=110, model__xgb__subsample=0.627489745430074; total time=
  37.1s
[CV] END model__ada__learning_rate=0.8259048896601802, model__ada__n_estimators=62, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=18, model__svc__
C=2.6873317142982747, model__xgb__learning_rate=0.34275258579123147, model__xgb__max_dep
th=7, model__xgb__n_estimators=110, model__xgb__subsample=0.627489745430074; total time=
  39.3s
[CV] END model__ada__learning_rate=0.8259048896601802, model__ada__n_estimators=62, mode
l__knn__n_neighbors=3, model__rf__max_depth=10, model__rf__n_estimators=18, model__svc__
C=2.6873317142982747, model__xgb__learning_rate=0.34275258579123147, model__xgb__max_dep
th=7, model__xgb__n_estimators=110, model__xgb__subsample=0.627489745430074; total time=
  35.4s
[CV] END model__ada__learning_rate=0.9112598890856902, model__ada__n_estimators=52, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=92, model__svc
__C=3.763424016750204, model__xgb__learning_rate=0.35525515120135054, model__xgb__max_de
pth=6, model__xgb__n_estimators=120, model__xgb__subsample=0.3190416561263947; total tim
e=  52.4s
[CV] END model__ada__learning_rate=0.9112598890856902, model__ada__n_estimators=52, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=92, model__svc
__C=3.763424016750204, model__xgb__learning_rate=0.35525515120135054, model__xgb__max_de
pth=6, model__xgb__n_estimators=120, model__xgb__subsample=0.3190416561263947; total tim
e=  55.2s
[CV] END model__ada__learning_rate=0.9112598890856902, model__ada__n_estimators=52, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=92, model__svc
__C=3.763424016750204, model__xgb__learning_rate=0.35525515120135054, model__xgb__max_de
pth=6, model__xgb__n_estimators=120, model__xgb__subsample=0.3190416561263947; total tim
e=  55.8s
[CV] END model__ada__learning_rate=0.9112598890856902, model__ada__n_estimators=52, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=92, model__svc
__C=3.763424016750204, model__xgb__learning_rate=0.35525515120135054, model__xgb__max_de
pth=6, model__xgb__n_estimators=120, model__xgb__subsample=0.3190416561263947; total tim
e=  52.8s
[CV] END model__ada__learning_rate=0.9112598890856902, model__ada__n_estimators=52, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=92, model__svc
__C=3.763424016750204, model__xgb__learning_rate=0.35525515120135054, model__xgb__max_de
pth=6, model__xgb__n_estimators=120, model__xgb__subsample=0.3190416561263947; total tim
e=  54.0s
[CV] END model__ada__learning_rate=0.87076585849603, model__ada__n_estimators=48, model_
_knn__n_neighbors=1, model__rf__max_depth=10, model__rf__n_estimators=97, model__svc__C=
9.805803133771734, model__xgb__learning_rate=0.34007750533977604, model__xgb__max_depth=
7, model__xgb__n_estimators=122, model__xgb__subsample=0.835471902316177; total time= 1.
6min
[CV] END model__ada__learning_rate=0.87076585849603, model__ada__n_estimators=48, model_
_knn__n_neighbors=1, model__rf__max_depth=10, model__rf__n_estimators=97, model__svc__C=
9.805803133771734, model__xgb__learning_rate=0.34007750533977604, model__xgb__max_depth=
7, model__xgb__n_estimators=122, model__xgb__subsample=0.835471902316177; total time= 1.
7min
[CV] END model__ada__learning_rate=0.87076585849603, model__ada__n_estimators=48, model_
_knn__n_neighbors=1, model__rf__max_depth=10, model__rf__n_estimators=97, model__svc__C=
9.805803133771734, model__xgb__learning_rate=0.34007750533977604, model__xgb__max_depth=
7, model__xgb__n_estimators=122, model__xgb__subsample=0.835471902316177; total time= 1.
7min
[CV] END model__ada__learning_rate=0.87076585849603, model__ada__n_estimators=48, model_
_knn__n_neighbors=1, model__rf__max_depth=10, model__rf__n_estimators=97, model__svc__C=
9.805803133771734, model__xgb__learning_rate=0.34007750533977604, model__xgb__max_depth=
7, model__xgb__n_estimators=122, model__xgb__subsample=0.835471902316177; total time= 1.
6min
[CV] END model__ada__learning_rate=0.87076585849603, model__ada__n_estimators=48, model_
_knn__n_neighbors=1, model__rf__max_depth=10, model__rf__n_estimators=97, model__svc__C=
9.805803133771734, model__xgb__learning_rate=0.34007750533977604, model__xgb__max_depth=
7, model__xgb__n_estimators=122, model__xgb__subsample=0.835471902316177; total time= 1.
7min
[CV] END model__ada__learning_rate=0.7538381698762668, model__ada__n_estimators=46, mode
l__knn__n_neighbors=2, model__rf__max_depth=5, model__rf__n_estimators=53, model__svc__C
=0.5567896524540005, model__xgb__learning_rate=0.39494603360716385, model__xgb__max_dept
h=7, model__xgb__n_estimators=87, model__xgb__subsample=0.6528003075654067; total time=
19.3s
[CV] END model__ada__learning_rate=0.7538381698762668, model__ada__n_estimators=46, mode
l__knn__n_neighbors=2, model__rf__max_depth=5, model__rf__n_estimators=53, model__svc__C
=0.5567896524540005, model__xgb__learning_rate=0.39494603360716385, model__xgb__max_dept
h=7, model__xgb__n_estimators=87, model__xgb__subsample=0.6528003075654067; total time=
20.6s
[CV] END model__ada__learning_rate=0.7538381698762668, model__ada__n_estimators=46, mode
l__knn__n_neighbors=2, model__rf__max_depth=5, model__rf__n_estimators=53, model__svc__C
=0.5567896524540005, model__xgb__learning_rate=0.39494603360716385, model__xgb__max_dept
h=7, model__xgb__n_estimators=87, model__xgb__subsample=0.6528003075654067; total time=
20.1s
[CV] END model__ada__learning_rate=0.7538381698762668, model__ada__n_estimators=46, mode
l__knn__n_neighbors=2, model__rf__max_depth=5, model__rf__n_estimators=53, model__svc__C
=0.5567896524540005, model__xgb__learning_rate=0.39494603360716385, model__xgb__max_dept
h=7, model__xgb__n_estimators=87, model__xgb__subsample=0.6528003075654067; total time=
19.4s
[CV] END model__ada__learning_rate=0.7538381698762668, model__ada__n_estimators=46, mode
l__knn__n_neighbors=2, model__rf__max_depth=5, model__rf__n_estimators=53, model__svc__C
=0.5567896524540005, model__xgb__learning_rate=0.39494603360716385, model__xgb__max_dept
h=7, model__xgb__n_estimators=87, model__xgb__subsample=0.6528003075654067; total time=
19.1s
[CV] END model__ada__learning_rate=0.9853119763020483, model__ada__n_estimators=43, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=32, model__svc
__C=7.384984962121942, model__xgb__learning_rate=0.20672608866707592, model__xgb__max_de
pth=8, model__xgb__n_estimators=77, model__xgb__subsample=0.5534191877236918; total time
= 1.2min
[CV] END model__ada__learning_rate=0.9853119763020483, model__ada__n_estimators=43, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=32, model__svc
__C=7.384984962121942, model__xgb__learning_rate=0.20672608866707592, model__xgb__max_de
pth=8, model__xgb__n_estimators=77, model__xgb__subsample=0.5534191877236918; total time
= 1.1min
[CV] END model__ada__learning_rate=0.9853119763020483, model__ada__n_estimators=43, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=32, model__svc
__C=7.384984962121942, model__xgb__learning_rate=0.20672608866707592, model__xgb__max_de
pth=8, model__xgb__n_estimators=77, model__xgb__subsample=0.5534191877236918; total time
= 1.0min
[CV] END model__ada__learning_rate=0.9853119763020483, model__ada__n_estimators=43, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=32, model__svc
__C=7.384984962121942, model__xgb__learning_rate=0.20672608866707592, model__xgb__max_de
pth=8, model__xgb__n_estimators=77, model__xgb__subsample=0.5534191877236918; total time
= 1.0min
[CV] END model__ada__learning_rate=0.9853119763020483, model__ada__n_estimators=43, mode
l__knn__n_neighbors=3, model__rf__max_depth=None, model__rf__n_estimators=32, model__svc
__C=7.384984962121942, model__xgb__learning_rate=0.20672608866707592, model__xgb__max_de
pth=8, model__xgb__n_estimators=77, model__xgb__subsample=0.5534191877236918; total time
= 1.1min
model__ada__learning_rate: 0.9853119763020483
model__ada__n_estimators: 43
model__knn__n_neighbors: 3
model__rf__max_depth: None
model__rf__n_estimators: 32
model__svc__C: 7.384984962121942
model__xgb__learning_rate: 0.20672608866707592
model__xgb__max_depth: 8
model__xgb__n_estimators: 77
model__xgb__subsample: 0.5534191877236918
```

In [43]:
```python
from sklearn.model_selection import cross_val_score
```

```
# Perform 5-fold cross validation
scores = cross_val_score(pipe, X_resampled, y_resampled, cv=5)

# Print the mean cross-validation score
print("Cross-Validation Mean Score:", scores.mean())
```

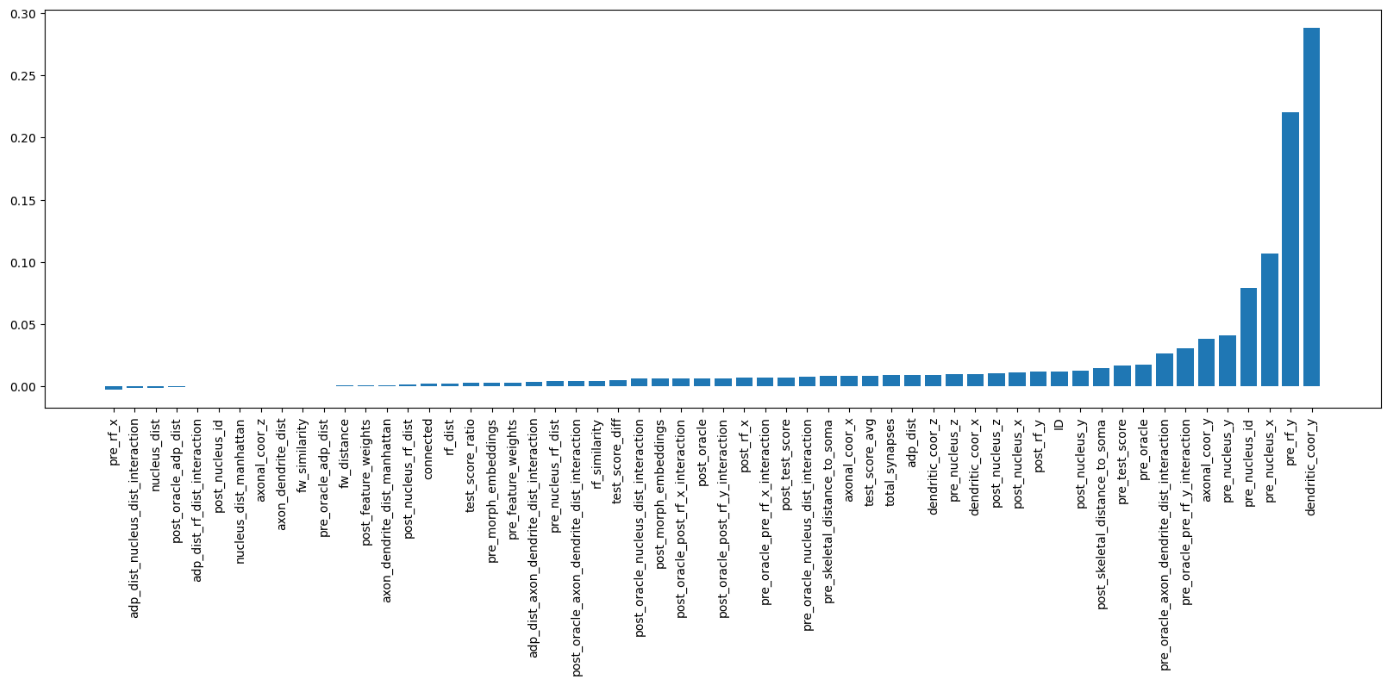
```
Cross-Validation Mean Score: 0.7726027397260273
```

Perform Permutation Importance

Calculates mean importance of each feature by repeatedly shuffling the values of each feature and measuring how much the model's accuracy decreases

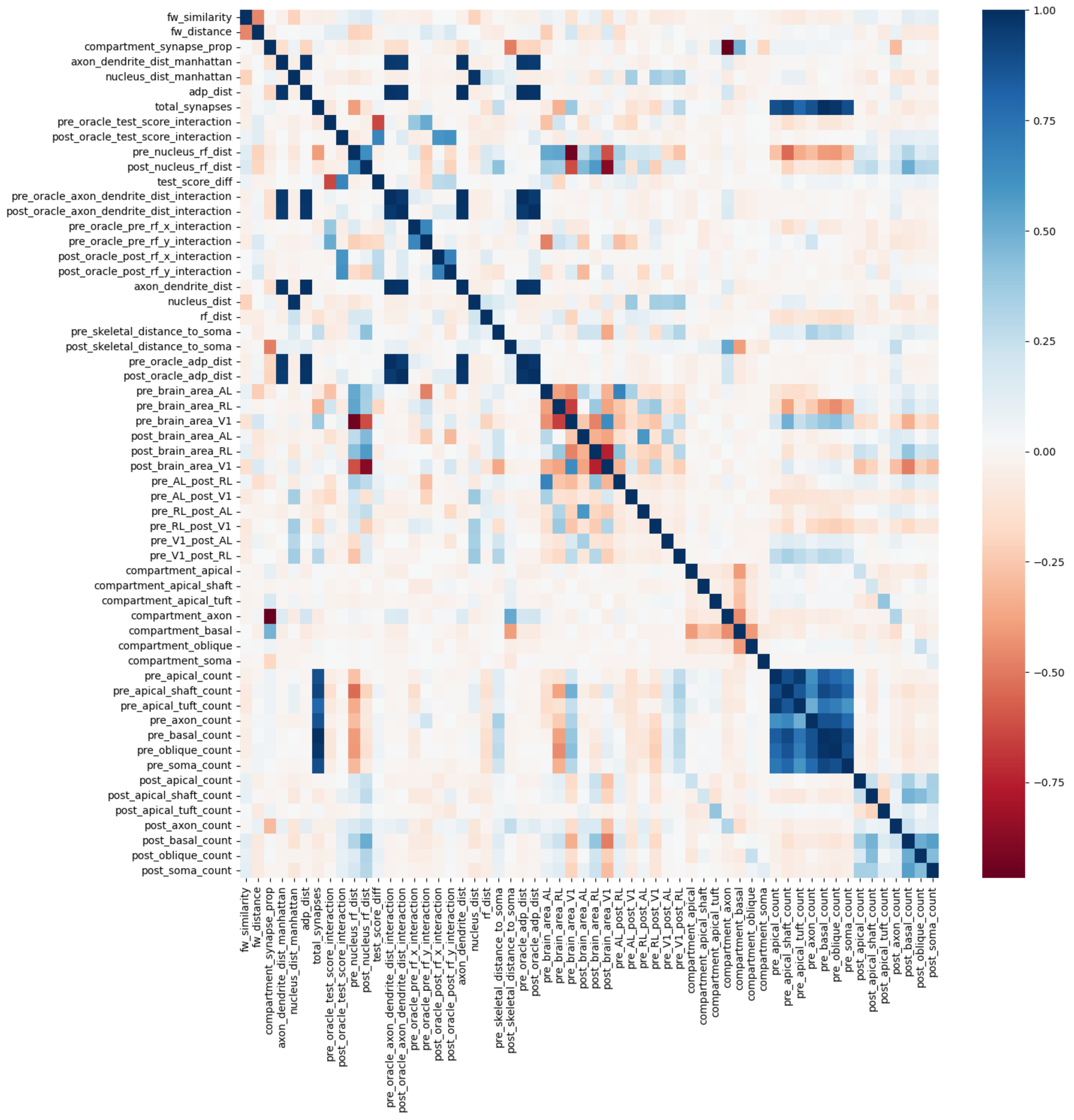
In [27]:
```
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt
import numpy as np

# perform permutation importance
results = permutation_importance(pipe, X_resampled, y_resampled, scoring='accuracy')

# get importance
importance = results.importances_mean

# get feature names
feature_names = train_data.columns.tolist()
```

In [28]:
```
# plot feature importance
plt.figure(figsize=(20, 6))  # Adjust the width and height of the figure
importance_sorted, feature_names_sorted = zip(*sorted(zip(importance, feature_names)))
plt.bar(feature_names_sorted, importance_sorted)
plt.xticks(rotation=90)  # Set rotation of the x-axis labels to 90 for vertical labels
plt.show()
```

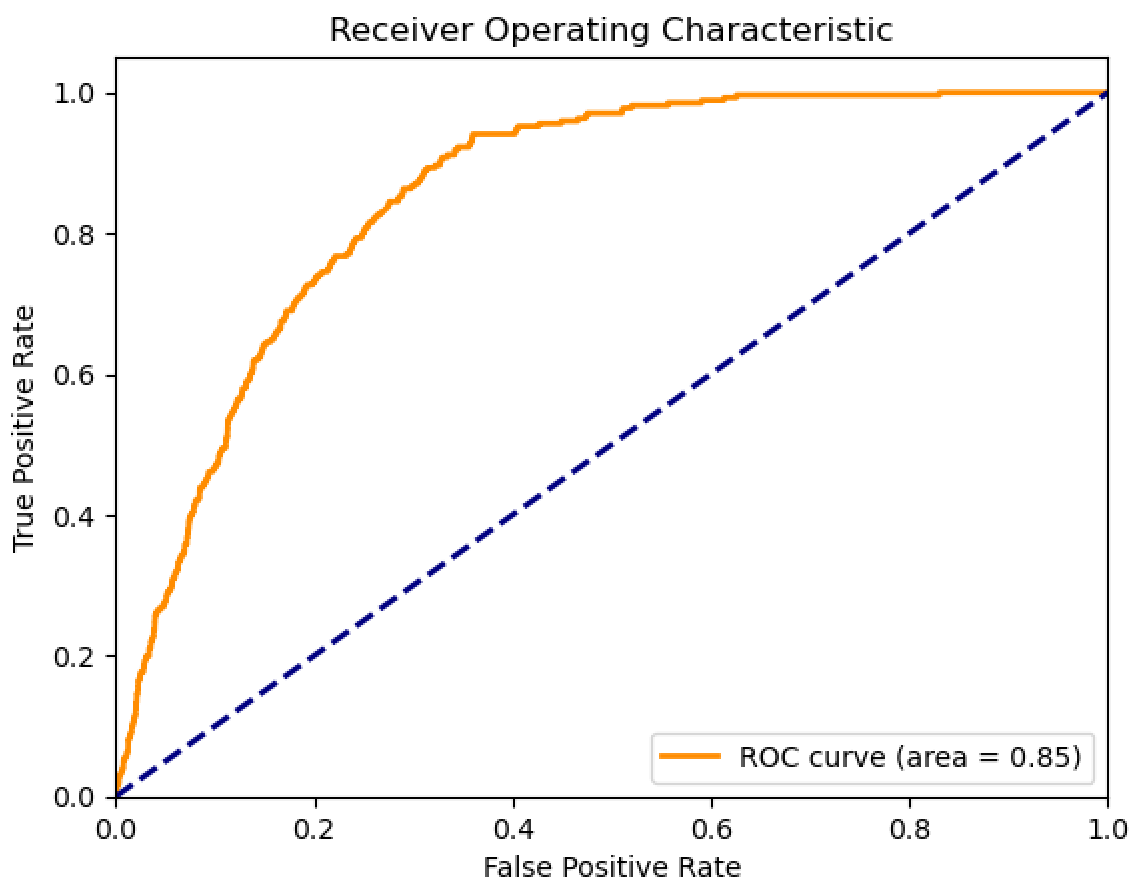


In [41]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# calculate correlation matrix
corr = X_resampled.corr()

# create a larger figure
```

```
plt.figure(figsize=(15, 15))
4
# create a heatmap
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cmap='RdBu')

# rotate x-axis labels for better readability
plt.xticks(rotation=90)

# show the plot
plt.show()
```



In [30]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(test_data['connected'], test_data['pred'])
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



A Potentially Better Model (After Competition Ended)

After the competition ended, using these base models with the following parameters yielded better accuracy scores than our best model (although not by a huge amount):

- **Linear SVM** W/ C=10
- **Logistic Regression** W/ default parameter values
- **Random Forest** W/ 10 Trees
- (Extreme) **Gradient Boosting** W/ A maximum tree depth of 7, learning rate of 0.2, and subsample of 0.5

and the final estimator was a **Linear SVM** with the same parameters as before.

Note: The model ran faster as well

Accuracies

We get a model accuracy of **~73.24%** and a balanced accuracy of **~78.1%**

After performing **5-fold** cross-validation, we get a mean CV score of **~77.306%**

Kaggle Results Unavailable As The Competition Has Ended

In [63]:
```
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import StackingClassifier, AdaBoostClassifier, GradientBoostingCla
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from imblearn.over_sampling import ADASYN
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler

# define base models
base_models = [
    ("svc", SVC(kernel='linear', C=1.0, probability=True, random_state=2)),  # C is the
    ("logreg", LogisticRegression(C=1.0, random_state=2)),  # C is the inverse of regula
    ("rf", RandomForestClassifier(random_state=1, n_estimators=10, bootstrap=True)),
    ("xgb", XGBClassifier(random_state=2, max_depth=7, learning_rate=0.2, n_estimators=1
]

# create pipeline
pipe = Pipeline(
    [("scaler", StandardScaler()),
     ("model", StackingClassifier(estimators=base_models, final_estimator=SVC(kernel='li
)

# undersample majority class
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(
    train_data[["fw_similarity", "fw_distance", "compartment_synapse_prop", "axon_dendri
    "compartment_apical",
    "compartment_apical_shaft",
    "compartment_apical_tuft",
    "compartment_axon",
    "compartment_basal",
    "compartment_oblique",
    "compartment_soma",
    "pre_apical_count",
    "pre_apical_shaft_count",
    "pre_apical_tuft_count",
    "pre_axon_count",
    "pre_basal_count",
    "pre_oblique_count",
    "pre_soma_count",
    "post_apical_count",
    "post_apical_shaft_count",
    "post_apical_tuft_count",
    "post_axon_count",
    "post_basal_count",
    "post_oblique_count",
    "post_soma_count"]],
    train_data["connected"]
)

# fit model
pipe.fit(X_resampled, y_resampled)

# predict on test data
test_data["pred"] = pipe.predict_proba(test_data[["fw_similarity", "fw_distance", "compa
    "compartment_apical",
    "compartment_apical_shaft",
    "compartment_apical_tuft",
    "compartment_axon",
    "compartment_basal",
    "compartment_oblique",
    "compartment_soma",
    "pre_apical_count",
    "pre_apical_shaft_count",
    "pre_apical_tuft_count",
    "pre_axon_count",
    "pre_basal_count",
    "pre_oblique_count",
    "pre_soma_count",
    "post_apical_count",
    "post_apical_shaft_count",
    "post_apical_tuft_count",
    "post_axon_count",
    "post_basal_count",
    "post_oblique_count",
    "post_soma_count"]])[:, 1]

# compute accuracy
print(f"accuracy: {accuracy_score(test_data['connected'], test_data['pred'] > .5)}")

# confusion matrix
print(confusion_matrix(test_data['connected'], test_data['pred'] > .5))

# compute balanced accuracy
print(
    f"balanced accuracy: {balanced_accuracy_score(test_data['connected'], test_data['pre
)
```

```
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
accuracy: 0.7323701132725267
[[26995  9901]
 [   46   225]]
balanced accuracy: 0.7809547150382605
```

In [64]:
```
from sklearn.model_selection import cross_val_score

# Perform 5-fold cross validation
scores = cross_val_score(pipe, X_resampled, y_resampled, cv=5)

# Print the mean cross-validation score
print("Cross-Validation Mean Score:", scores.mean())
```

```
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
```

```
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\micha\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Cross-Validation Mean Score: 0.7730593607305936
```

In [31]:
```
print("Main Dataframe Size:",data.shape)
print("\nMain Dataframe:")
data.info()
```

```
Main Dataframe Size: (185832, 109)

Main Dataframe:
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 185832 entries, 0 to 185831
Columns: 109 entries, ID to synapse_ratio
dtypes: bool(2), float64(56), int64(20), object(5), uint8(26)
memory usage: 121.2+ MB
```

Leaderboard Data

From here we have the code used to create the csv file used in Kaggle

In [63]:
```
#we need to first load and merge the leaderboard data to have the same format as the tra
lb_data = pd.read_csv("./leaderboard_data.csv")
```

In [64]:
```
lb_data = (
    lb_data.merge(
        feature_weights.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        feature_weights.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
)
```

In [30]:
```
print(lb_data.columns)
```

```
Index(['ID', 'axonal_coor_x', 'axonal_coor_y', 'axonal_coor_z',
       'dendritic_coor_x', 'dendritic_coor_y', 'dendritic_coor_z', 'adp_dist',
       'post_skeletal_distance_to_soma', 'pre_skeletal_distance_to_soma',
       'pre_oracle', 'pre_test_score', 'pre_rf_x', 'pre_rf_y', 'post_oracle',
       'post_test_score', 'post_rf_x', 'post_rf_y', 'compartment',
       'pre_brain_area', 'post_brain_area', 'pre_nucleus_x', 'pre_nucleus_y',
       'pre_nucleus_z', 'post_nucleus_x', 'post_nucleus_y', 'post_nucleus_z',
       'pre_nucleus_id', 'post_nucleus_id', 'pre_feature_weights',
       'post_feature_weights', 'pre_morph_embeddings',
       'post_morph_embeddings'],
      dtype='object')
```

In [65]:
```
# compute the cosine similarity between the pre- and post- feature weights
lb_data["fw_similarity"] = lb_data.apply(row_feature_similarity, axis=1)

lb_data["fw_distance"] = lb_data.apply(row_feature_distance, axis=1)

# Create a copy of the 'compartment' column
```

```
lb_data['compartment_copy'] = lb_data['compartment']

# Map the proportions to the original DataFrame
lb_data['compartment_synapse_prop'] = lb_data['compartment_copy'].map(synapse_proportion

# Compute Euclidean distance for leaderboard data
lb_data['axon_dendrite_dist'] = np.sqrt(
    (lb_data['axonal_coor_x'] - lb_data['dendritic_coor_x'])**2 +
    (lb_data['axonal_coor_y'] - lb_data['dendritic_coor_y'])**2 +
    (lb_data['axonal_coor_z'] - lb_data['dendritic_coor_z'])**2
)
lb_data['nucleus_dist'] = np.sqrt(
    (lb_data['pre_nucleus_x'] - lb_data['post_nucleus_x'])**2 +
    (lb_data['pre_nucleus_y'] - lb_data['post_nucleus_y'])**2 +
    (lb_data['pre_nucleus_z'] - lb_data['post_nucleus_z'])**2
)

# Compute interaction between oracle and distance features for train data
lb_data['pre_oracle_adp_dist'] = lb_data['pre_oracle'] * lb_data['adp_dist']
lb_data['post_oracle_adp_dist'] = lb_data['post_oracle'] * lb_data['adp_dist']

# Compute Euclidean distance between receptive field locations for train data
lb_data['rf_dist'] = np.sqrt(
    (lb_data['pre_rf_x'] - lb_data['post_rf_x'])**2 +
    (lb_data['pre_rf_y'] - lb_data['post_rf_y'])**2
)

lb_data = pd.get_dummies(lb_data, columns=['pre_brain_area', 'post_brain_area'])

# Perform one-hot encoding on the 'compartment' column
lb_data = pd.get_dummies(lb_data, columns=['compartment'])

# Create interaction features
lb_data['pre_AL_post_RL'] = lb_data['pre_brain_area_AL'] * lb_data['post_brain_area_RL']
lb_data['pre_AL_post_V1'] = lb_data['pre_brain_area_AL'] * lb_data['post_brain_area_V1']
lb_data['pre_RL_post_AL'] = lb_data['pre_brain_area_RL'] * lb_data['post_brain_area_AL']
lb_data['pre_RL_post_V1'] = lb_data['pre_brain_area_RL'] * lb_data['post_brain_area_V1']
lb_data['pre_V1_post_AL'] = lb_data['pre_brain_area_V1'] * lb_data['post_brain_area_AL']
lb_data['pre_V1_post_RL'] = lb_data['pre_brain_area_V1'] * lb_data['post_brain_area_RL']

lb_data['pre_skeletal_distance_to_soma']
lb_data['pre_skeletal_distance_to_soma']

# Compute the Euclidean distance between the pre- and post- morphological embeddings
lb_data["morph_distance"] = lb_data.apply(row_morph_distance, axis=1)

# Fill NaNs with the mean of the column
mean_morph_distance = lb_data['morph_distance'].mean()
lb_data['morph_distance'].fillna(mean_morph_distance, inplace=True)

# Drop rows with NaNs in the 'morph_distance' column
lb_data.dropna(subset=['morph_distance'], inplace=True)

lb_data['pre_oracle_pre_rf_x_interaction'] = lb_data['pre_oracle'] * lb_data['pre_rf_x']
lb_data['pre_oracle_pre_rf_y_interaction'] = lb_data['pre_oracle'] * lb_data['pre_rf_y']
lb_data['post_oracle_post_rf_x_interaction'] = lb_data['post_oracle'] * lb_data['post_rf
lb_data['post_oracle_post_rf_y_interaction'] = lb_data['post_oracle'] * lb_data['post_rf

# Compute interaction between pre_oracle and axon_dendrite_dist
lb_data['pre_oracle_axon_dendrite_dist_interaction'] = lb_data['pre_oracle'] * lb_data['
lb_data['post_oracle_axon_dendrite_dist_interaction'] = lb_data['post_oracle'] * lb_data

# Compute interaction between post_oracle and nucleus_dist
# lb_data['pre_oracle_nucleus_dist_interaction'] = lb_data['pre_oracle'] * lb_data['nucl
# lb_data['post_oracle_nucleus_dist_interaction'] = lb_data['post_oracle'] * lb_data['nu
```

```
lb_data['test_score_diff'] = lb_data['post_test_score'] - lb_data['pre_test_score']

# Compute Euclidean distance between nucleus and receptive field locations
lb_data['pre_nucleus_rf_dist'] = np.sqrt(
    (lb_data['pre_nucleus_x'] - lb_data['pre_rf_x'])**2 +
    (lb_data['pre_nucleus_y'] - lb_data['pre_rf_y'])**2
)
lb_data['post_nucleus_rf_dist'] = np.sqrt(
    (lb_data['post_nucleus_x'] - lb_data['post_rf_x'])**2 +
    (lb_data['post_nucleus_y'] - lb_data['post_rf_y'])**2
)

# Compute interaction between oracle and test score
lb_data['pre_oracle_test_score_interaction'] = lb_data['pre_oracle'] * lb_data['pre_test
lb_data['post_oracle_test_score_interaction'] = lb_data['post_oracle'] * lb_data['post_t

# Compute Manhattan distance
lb_data['axon_dendrite_dist_manhattan'] = (
    abs(lb_data['axonal_coor_x'] - lb_data['dendritic_coor_x']) +
    abs(lb_data['axonal_coor_y'] - lb_data['dendritic_coor_y']) +
    abs(lb_data['axonal_coor_z'] - lb_data['dendritic_coor_z'])
)
lb_data['nucleus_dist_manhattan'] = (
    abs(lb_data['pre_nucleus_x'] - lb_data['post_nucleus_x']) +
    abs(lb_data['pre_nucleus_y'] - lb_data['post_nucleus_y']) +
    abs(lb_data['pre_nucleus_z'] - lb_data['post_nucleus_z'])
)

# Calculate the total number of synapses for each neuron
total_synapses = lb_data.groupby('pre_nucleus_id').size()

# Add the Total Synapses to your DataFrame
lb_data['total_synapses'] = lb_data['pre_nucleus_id'].map(total_synapses)

# List of all compartment types
compartments = ['compartment_apical', 'compartment_apical_shaft', 'compartment_apical_tu

# Create a new feature that represents the total count of compartments for each neuron
lb_data['total_compartments'] = lb_data[compartments].sum(axis=1)

# Create new features that represent the ratio of each compartment type to the total num
for compartment in compartments:
    lb_data[f'{compartment}_ratio'] = lb_data[compartment] / lb_data['total_compartments

# List of all compartment types
compartments = ['apical', 'apical_shaft', 'apical_tuft', 'axon', 'basal', 'oblique', 'so

# Create new aggregated features for pre_nucleus_id
for compartment in compartments:
    # Create new feature name
    new_feature = f'pre_{compartment}_count'

    # Create new feature
    lb_data[new_feature] = lb_data.groupby('pre_nucleus_id')[f'compartment_{compartment}

# Create new aggregated features for post_nucleus_id
for compartment in compartments:
    # Create new feature name
    new_feature = f'post_{compartment}_count'

    # Create new feature
    lb_data[new_feature] = lb_data.groupby('post_nucleus_id')[f'compartment_{compartment
```

In [66]:
```
# fit model
pipe.fit(X_resampled, y_resampled)
```

```
# predict on leaderboard data
lb_data["pred"] = pipe.predict_proba(lb_data[["fw_similarity", "fw_distance", "compartme

# create a boolean prediction solution
lb_data["connected"] = lb_data["pred"] > .5
```

In [67]:
```
# columns should be ID, connected
submission_data = lb_data.filter(['ID','connected'])
```

In [68]:
```
#writing csv files
submission_data.to_csv('submission_data_19_RF2.csv',index=False)
```

In [ ]:
```

```