```
import pandas as pd

data =
pd.read_csv('https://raw.githubusercontent.com/npradaschnor/Pima-
Indians-Diabetes-Dataset/master/diabetes.csv')

data.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0            6      148             72             35        0  33.6

1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
X = data.drop('Outcome', axis=1)
y = data['Outcome']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=2)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=1000)
```

```
y_scores = model.predict_proba(X_test)[:,1]



y_scores
```

```
array([0.2758503 , 0.18842069, 0.11448981, 0.16368795, 0.4715251 ,
       0.44170194, 0.01545458, 0.6605503 , 0.54136875, 0.77717331,
       0.25601769, 0.89728056, 0.33614441, 0.30308996, 0.08199125,
```

```
       0.38084702, 0.13934778, 0.07482402, 0.86665798, 0.5613247 ,
       0.20835836, 0.07736815, 0.53982239, 0.09456509, 0.53967271,
       0.88554708, 0.1243389 , 0.03014779, 0.25152025, 0.11579709,
       0.91109793, 0.8706549 , 0.76531008, 0.83743011, 0.6165898 ,
       0.68350024, 0.96806648, 0.24353318, 0.51154452, 0.73599347,
       0.06983745, 0.59378797, 0.58351287, 0.32753119, 0.02760688,
       0.50239856, 0.63926574, 0.22539934, 0.36059905, 0.95624662,
       0.04890383, 0.66102112, 0.81174894, 0.24555107, 0.09320651,
       0.04152459, 0.77799846, 0.00570139, 0.40866977, 0.75690546,
       0.7413723 , 0.35188999, 0.19230059, 0.20514197, 0.0768586 ,
       0.6270294 , 0.050907  , 0.73293779, 0.03690838, 0.71583657,
       0.67525971, 0.07016837, 0.18115086, 0.11426867, 0.09119426,
       0.51835688, 0.16355232, 0.13695503, 0.13173048, 0.2341894 ,
       0.6556617 , 0.1468054 , 0.06144124, 0.37496999, 0.25806706,
       0.83587578, 0.90265268, 0.30266343, 0.12370914, 0.08538068,
       0.06547828, 0.23718831, 0.00417029, 0.55038102, 0.51677531,
       0.64959804, 0.36889176, 0.12823988, 0.60576647, 0.08216269,
       0.72400009, 0.06248435, 0.77818672, 0.50287874, 0.64210666,
       0.21914482, 0.25727444, 0.73553291, 0.12705332, 0.53432103,
       0.0987172 , 0.31806545, 0.01994664, 0.73542507, 0.17451194,
       0.3389918 , 0.77362628, 0.21918291, 0.06360321, 0.58429026,
       0.0585205 , 0.29639387, 0.2408059 , 0.07700151, 0.26486643,
       0.42037419, 0.0404956 , 0.87107584, 0.97118111, 0.72966335,
       0.70244958, 0.85198486, 0.08457216, 0.42031934, 0.81228644,
       0.11171403, 0.15441586, 0.85444161, 0.80094282, 0.01302516,
       0.0948891 , 0.03977236, 0.20215332, 0.37668596, 0.12403437,
       0.29125775, 0.14051793, 0.02145003, 0.43005226, 0.75090476,
       0.11865543, 0.48561671, 0.22505638, 0.2000772 ])
```

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_scores)

thresholds
```

```
array([1.97118111, 0.97118111, 0.96806648, 0.95624662, 0.90265268,
       0.89728056, 0.77818672, 0.77717331, 0.73293779, 0.72400009,
       0.68350024, 0.67525971, 0.66102112, 0.6605503 , 0.6556617 ,
       0.64959804, 0.63926574, 0.6165898 , 0.60576647, 0.58351287,
       0.55038102, 0.53967271, 0.53432103, 0.50287874, 0.50239856,
       0.42037419, 0.42031934, 0.40866977, 0.38084702, 0.3389918 ,
       0.32753119, 0.30308996, 0.30266343, 0.25727444, 0.2408059 ,
       0.22539934, 0.22505638, 0.20835836, 0.20514197, 0.16355232,
       0.14051793, 0.12823988, 0.12705332, 0.11448981, 0.11426867,
       0.04152459, 0.0404956 , 0.00417029])
```

```
import plotly.graph_objects as go
import numpy as np
```

```
# Generate a trace for ROC curve
trace0 = go.Scatter(
    x=fpr,
    y=tpr,
    mode='lines',
    name='ROC curve'
)

# Only label every nth point to avoid cluttering
n = 10
indices = np.arange(len(thresholds)) % n == 0  # Choose indices where
index mod n is 0

trace1 = go.Scatter(
    x=fpr[indices],
    y=tpr[indices],
    mode='markers+text',
    name='Threshold points',
    text=[f"Thr={thr:.2f}" for thr in thresholds[indices]],
    textposition='top center'
)


# Diagonal line
trace2 = go.Scatter(
    x=[0, 1],
    y=[0, 1],
    mode='lines',
    name='Random (Area = 0.5)',
    line=dict(dash='dash')
)

data = [trace0, trace1, trace2]

# Define layout with square aspect ratio
layout = go.Layout(
    title='Receiver Operating Characteristic',
    xaxis=dict(title='False Positive Rate'),
    yaxis=dict(title='True Positive Rate'),
    autosize=False,
    width=800,
    height=800,
    showlegend=False
)

# Define figure and add data
fig = go.Figure(data=data, layout=layout)

# Show figure
fig.show()
```

```python
# Assume that fpr, tpr, thresholds have already been calculated
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Optimal threshold is:", optimal_threshold)
```

```
Optimal threshold is: 0.5503810234218872
```

```python
import plotly.graph_objects as go
import numpy as np
from sklearn.metrics import roc_auc_score

# Assuming fpr, tpr, thresholds are already calculated as before
fpr, tpr, thresholds = roc_curve(y_test, y_scores)

# Calculate the AUC (Area Under the Curve)
roc_auc = roc_auc_score(y_test, y_scores)

# Generate a trace for ROC curve
trace0 = go.Scatter(
    x=fpr,
    y=tpr,
    mode='lines',
    name=f'ROC curve (Area = {roc_auc:.2f})'
)

# Only label every nth point to avoid cluttering
n = 10
indices = np.arange(len(thresholds)) % n == 0  # Choose indices where
index mod n is 0

trace1 = go.Scatter(
    x=fpr[indices],
    y=tpr[indices],
    mode='markers+text',
    name='Threshold points',
    text=[f"Thr={thr:.2f}" for thr in thresholds[indices]],
    textposition='top center'
)

# Diagonal line
trace2 = go.Scatter(
    x=[0, 1],
    y=[0, 1],
    mode='lines',
    name='Random (Area = 0.5)',
    line=dict(dash='dash')
)

data = [trace0, trace1, trace2]
```

```
# Define layout with square aspect ratio
layout = go.Layout(
    title='Receiver Operating Characteristic',
    xaxis=dict(title='False Positive Rate'),
    yaxis=dict(title='True Positive Rate'),
    autosize=False,
    width=800,
    height=800,
    showlegend=True
)

# Define figure and add data
fig = go.Figure(data=data, layout=layout)

# Show figure
fig.show()



import numpy as np
import plotly.graph_objects as go
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.preprocessing import StandardScaler

# Assuming that X_train, X_test, y_train, y_test are already defined

# SVM requires feature scaling for better performance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Logistic Regression model
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
lr_scores = lr_model.predict_proba(X_test)[:,1]

# SVM model
svm_model = SVC(probability=True)
svm_model.fit(X_train_scaled, y_train)
svm_scores = svm_model.predict_proba(X_test_scaled)[:,1]

# Generate ROC curve data for logistic regression model
lr_fpr, lr_tpr, lr_thresholds = roc_curve(y_test, lr_scores)
lr_auc = roc_auc_score(y_test, lr_scores)

# Generate ROC curve data for SVM model
svm_fpr, svm_tpr, svm_thresholds = roc_curve(y_test, svm_scores)
svm_auc = roc_auc_score(y_test, svm_scores)
```

```
# Generate a trace for the Logistic Regression ROC curve
trace0 = go.Scatter(
    x=lr_fpr,
    y=lr_tpr,
    mode='lines',
    name=f'Logistic Regression (Area = {lr_auc:.2f})'
)

# Generate a trace for the SVM ROC curve
trace1 = go.Scatter(
    x=svm_fpr,
    y=svm_tpr,
    mode='lines',
    name=f'SVM (Area = {svm_auc:.2f})'
)

# Diagonal line
trace2 = go.Scatter(
    x=[0, 1],
    y=[0, 1],
    mode='lines',
    name='Random (Area = 0.5)',
    line=dict(dash='dash')
)

data = [trace0, trace1, trace2]

# Define layout with square aspect ratio
layout = go.Layout(
    title='Receiver Operating Characteristic',
    xaxis=dict(title='False Positive Rate'),
    yaxis=dict(title='True Positive Rate'),
    autosize=False,
    width=800,
    height=800,
    showlegend=True
)

# Define figure and add data
fig = go.Figure(data=data, layout=layout)

# Show figure
fig.show()
```