

```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Page configuration
st.set_page_config(layout="wide")
st.title('Iris Dataset Analysis')

# Load and prepare data
@st.cache_data
def load_data():
    iris = load_iris()
    df = pd.DataFrame(iris.data, columns=iris.feature_names)
    df['Species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
    return df, iris.feature_names

df, feature_names = load_data()
X = df[feature_names].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Sidebar controls
st.sidebar.header('Analysis Controls')
# ADDED: Slider for selecting the number of clusters
clusters = st.sidebar.slider("Select Number of Clusters", min_value=2, max_value=10,
                             value=3)

# 1. Feature Distribution Analysis
st.header('1. Feature Distributions by Species')

# Colors for species
colors = {'setosa': '#FF4B4B', 'versicolor': '#4B4BFF', 'virginica': '#4BFF4B'}

# Feature selection for box plot
_, col, _ = st.columns([1,3,1])
with col:
    selected_feature = st.selectbox('Select Feature for Box Plot:', feature_names)

# Create box plot for selected feature
fig_box = px.box(df,
                  y=selected_feature,
                  x='Species',
                  color='Species',
```

```

color_discrete_map=colors,
title=f'Distribution of {selected_feature} by Species',
labels={selected_feature: selected_feature, 'Species': 'Species'},
category_orders={'Species': sorted(dff['Species'].unique())}) # Sort legend

fig_box.update_layout(
    title=f'Distribution of {selected_feature} by Species',
    yaxis_title=selected_feature,
    showlegend=True
)

_, col, _ = st.columns([1,3,1])
with col:
    st.plotly_chart(fig_box)

# 2. Feature Relationships
st.header('2. Feature Relationships')
# ADDED: Scatter matrix plot
fig_scatter_matrix = px.scatter_matrix(
    df,
    dimensions=feature_names,
    color="Species",
    color_discrete_map=colors,
    title="Feature Relationships by Species"
)
st.plotly_chart(fig_scatter_matrix, use_container_width=True)

# 3. Feature Correlations
st.header('3. Feature Correlations')
correlation = df[feature_names].corr()

# Create correlation heatmap
fig_corr = go.Figure(data=go.Heatmap(
    z=correlation,
    x=feature_names,
    y=feature_names,
    colorscale='RdBu',
    zmin=-1,
    zmax=1,
    text=np.round(correlation, 2),
    texttemplate='%{text}',
    textfont={"size": 12},
    hoverongaps=False))

fig_corr.update_layout(
    title='Feature Correlation Matrix'
)

_, col, _ = st.columns([1,2,1])
with col:

```

```

st.plotly_chart(fig_corr)

# 4. Elbow Analysis
st.header('4. Elbow Analysis')
@st.cache_data
def perform_elbow_analysis(X, max_clusters=10):
    inertias = []
    for k in range(1, max_clusters + 1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X)
        inertias.append(kmeans.inertia_)
    return inertias

inertias = perform_elbow_analysis(X_scaled)

# ADDED: Line chart for Elbow Method
elbow_df = pd.DataFrame({'Number of Clusters': range(1, 11), 'Inertia': inertias})
fig_elbow = px.line(
    elbow_df,
    x='Number of Clusters',
    y='Inertia',
    markers=True,
    title='Elbow Method Analysis'
)
st.plotly_chart(fig_elbow, use_container_width=True)

# 5. Clustering Analysis
st.header('5. Clustering Analysis')

# Perform clustering (using the dynamically selected 'clusters' from sidebar)
kmeans = KMeans(n_clusters=clusters, random_state=42)
cluster_labels = kmeans.fit_predict(X_scaled)
df['Cluster'] = cluster_labels.astype(str)

# Create comparison plots
col1, col2 = st.columns(2)

with col1:
    st.subheader('Clustering Result')
    # ADDED: Scatter plot for KMeans result
    fig_cluster = px.scatter(
        df,
        x='petal length (cm)',
        y='petal width (cm)',
        color='Cluster',
        title='KMeans Clustering Result'
    )
    st.plotly_chart(fig_cluster, use_container_width=True)

with col2:

```

```

st.subheader('Actual Species')
# ADDED: Scatter plot for Actual Species
fig_actual = px.scatter(
    df,
    x='petal length (cm)',
    y='petal width (cm)',
    color='Species',
    color_discrete_map=colors,
    title='Actual Species Distribution'
)
st.plotly_chart(fig_actual, use_container_width=True)

# 6. Clustering Performance Analysis
st.header('6. Clustering Performance')
confusion_df = pd.crosstab(df['Species'], df['Cluster'], margins=True)
st.write("Confusion Matrix (Species vs Clusters):")
st.write(confusion_df)

# 7. Additional Statistics
st.header('7. Feature Statistics')
col3, col4 = st.columns(2)

with col3:
    st.subheader('Statistics by Species')
    species_stats = df.groupby('Species', observed=True)[feature_names].agg(['mean', 'std']).round(2)
    st.write(species_stats)

with col4:
    st.subheader('Statistics by Cluster')
    cluster_stats = df.groupby('Cluster', observed=True)[feature_names].agg(['mean', 'std']).round(2)
    st.write(cluster_stats)

```





