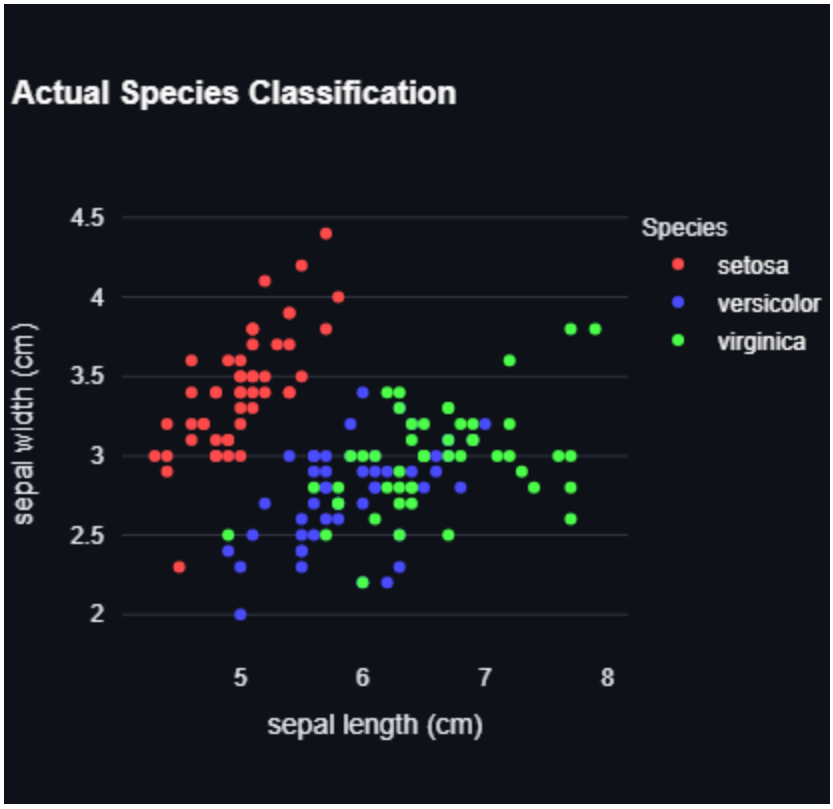
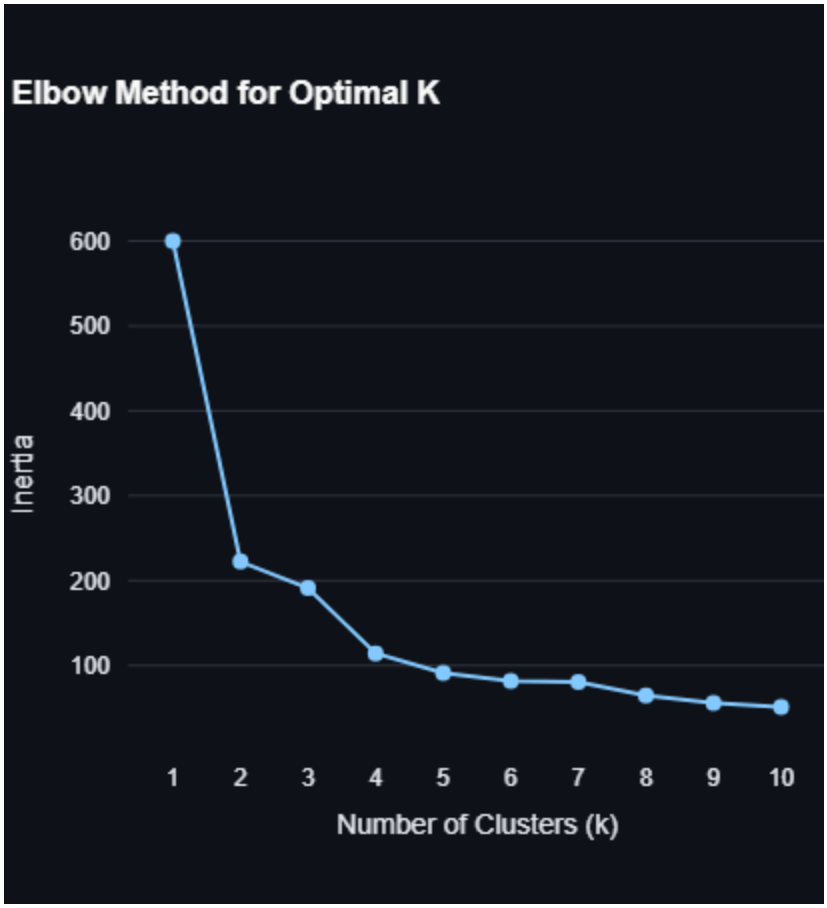
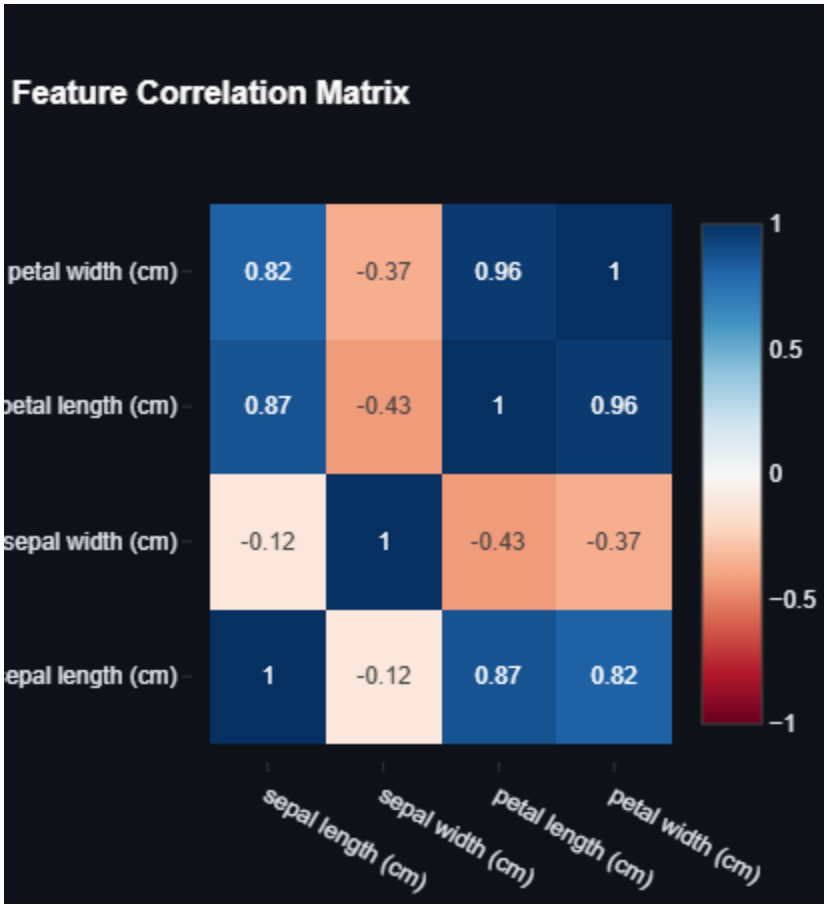
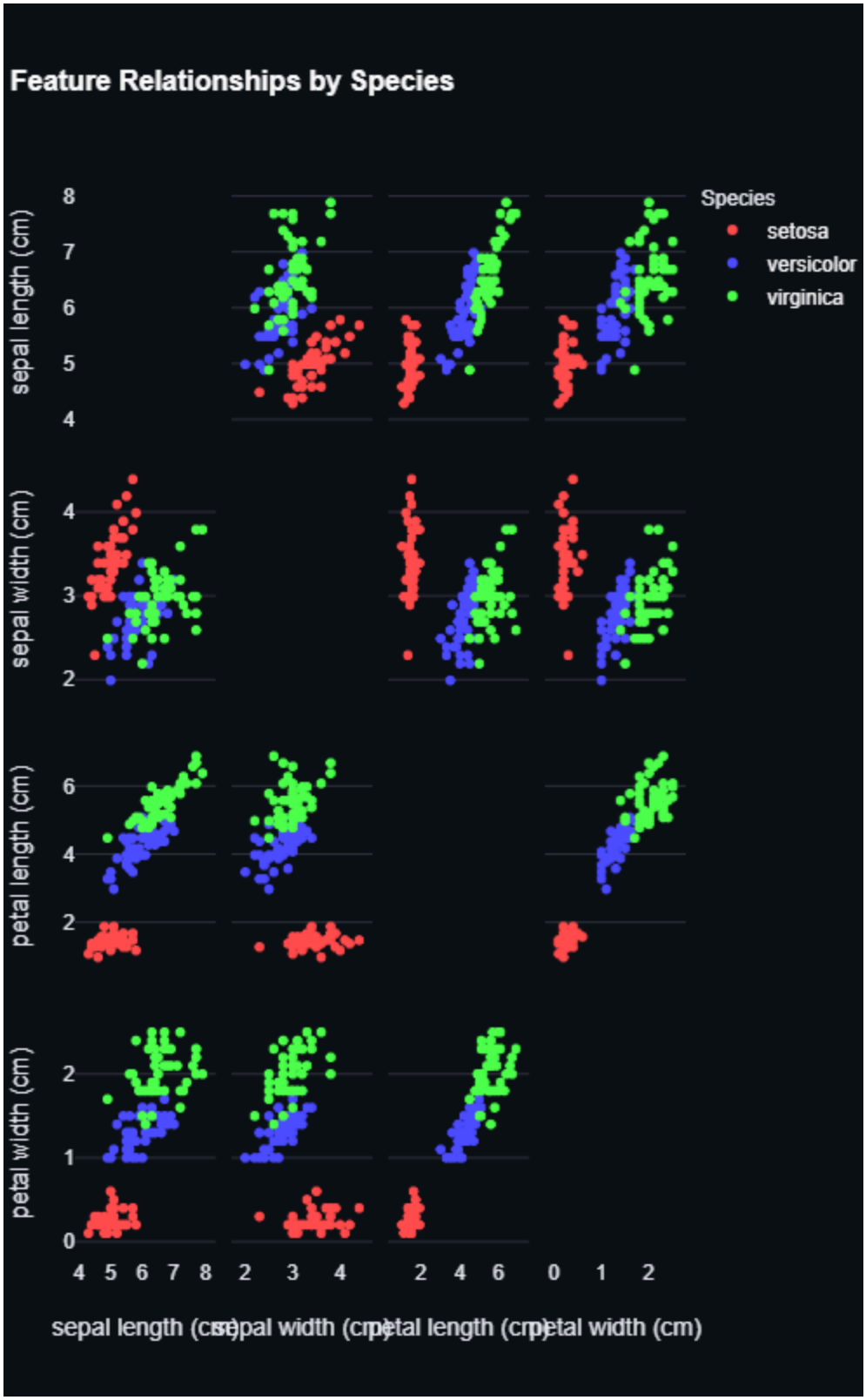
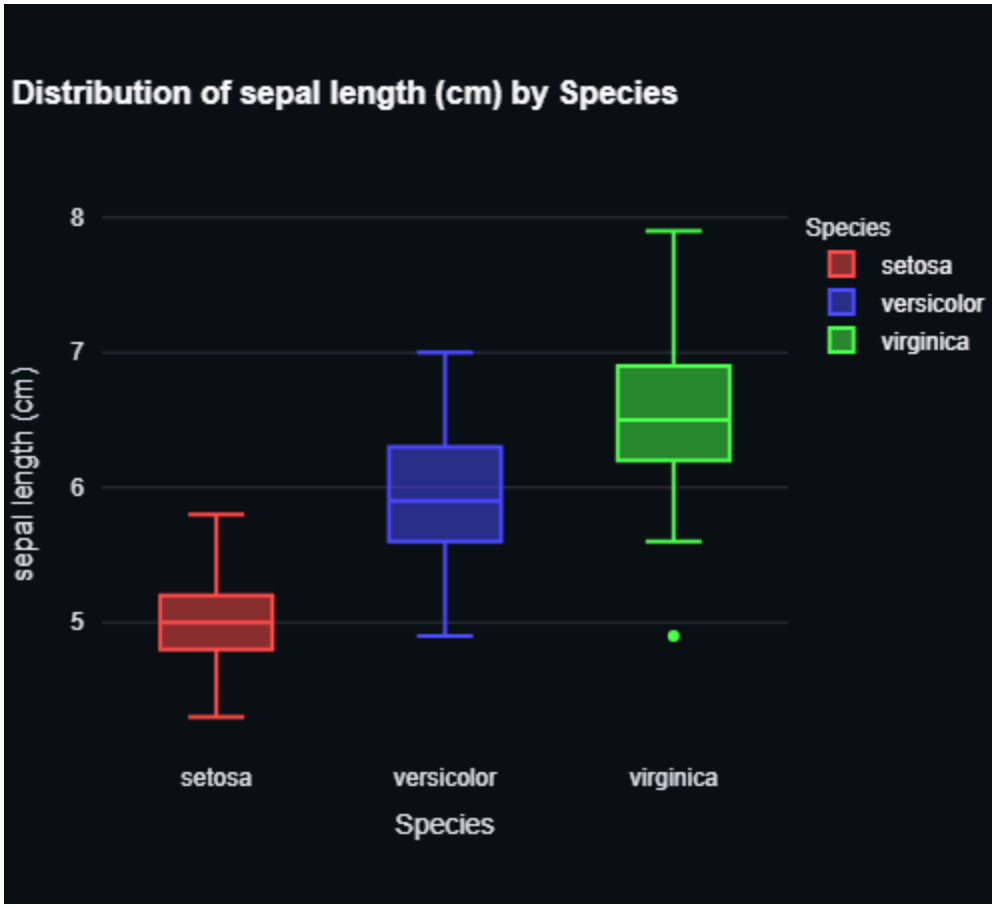

Actual Species Classification
4.5
4
3.5
3
2.5
2
sepal width (cm)
5
6
7
8
sepal length (cm)
Species
setosa
versicolor
virginica


KMeans Clustering Results
4.5
4
3.5
3
2.5
2
sepal width (cm)
5
6
7
8
sepal length (cm)
Cluster
1
2
4
0
3


Elbow Method for Optimal K
600
500
400
300
200
100
Inertia
1
2
3
4
5
6
7
8
9
10
Number of Clusters (k)


Feature Correlation Matrix
petal width (cm)
etal length (cm)
sepal width (cm)
epal length (cm)
0.82
-0.37
0.96
1
0.87
-0.43
1
0.96
-0.12
1
-0.43
-0.37
1
-0.12
0.87
0.82
1
0.5
0
−0.5
−1
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)


Feature Relationships by Species
Species
setosa
versicolor
virginica
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)




```
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')
n_clusters = st.sidebar.slider('Number of Clusters for KMeans:',
                               min_value=2,
                               max_value=6,
                               value=3,
                               step=1)

# 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(df['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')

# Create scatter matrix
fig_scatter = px.scatter_matrix(
    df,
    dimensions=feature_names,
    color='Species',
    color_discrete_map=colors,
    title='Feature Relationships by Species',
    height=800
)

fig_scatter.update_traces(diagonal_visible=False)
fig_scatter.update_layout(
    showlegend=True,
    dragmode='select'
)

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

# 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)

# Create elbow plot
fig_elbow = go.Figure()
fig_elbow.add_trace(
    go.Scatter(
```

```
        x=list(range(1, 11)),
        y=inertias,
        mode='lines+markers',
        marker=dict(size=8),
        line=dict(width=2)
    )
)

fig_elbow.update_layout(
    title='Elbow Method for Optimal K',
    xaxis_title='Number of Clusters (k)',
    yaxis_title='Inertia',
    showlegend=False,
    xaxis=dict(tickmode='linear', tick0=1, dtick=1)
)

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

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

# Perform clustering with selected number of clusters
kmeans = KMeans(n_clusters=n_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')
    fig_cluster = px.scatter(
        df,
        x=feature_names[0],
        y=feature_names[1],
        color='Cluster',
        title='KMeans Clustering Results',
        color_discrete_sequence=px.colors.qualitative.Plotly,
```

```
        height=400
    )
    st.plotly_chart(fig_cluster)

with col2:
    st.subheader('Actual Species')
    fig_species = px.scatter(
        df,
        x=feature_names[0],
        y=feature_names[1],
        color='Species',
        title='Actual Species Classification',
        color_discrete_map=colors,
        height=400
    )
    st.plotly_chart(fig_species)

# 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)
```