Deep Learning Models Link:

https://chatgpt.com/share/538ccffa-5122-412d-9074-3c934b728124

StreamLit Link:

https://chatgpt.com/share/c46656fd-492a-427e-b21f-e89461c30177

Try to made some changes but still there is warning in the code

```
import streamlit as st
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import ConfusionMatrixDisplay, roc_curve, precision_recall_curve
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix

def main():
    st.title("Binary Classification WebApp")
    st.markdown("Are your mushroom edible or poisonous? 🍄")

    st.sidebar.title("Binary Classification")
    st.sidebar.markdown("Are your mushroom edible or poisonous?")

    @st.cache(persist = True)
    def load_data():
        data =
pd.read_csv('https://raw.githubusercontent.com/pcsingh/ML-WebApp-with-Streamlit-and-Python
/master/mushrooms.csv')
        label = LabelEncoder()

        for col in data.columns:
            data[col] = label.fit_transform(data[col])

        return data

    @st.cache(persist = True)
    def split(df):
        y = df.type
        x = df.drop(columns = ['type'])
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)
```

```
        return x_train, x_test, y_train, y_test

    def plot_metrics(metrics_list):

        if 'Confusion Matrix' in metrics_list:
            st.subheader("Confusion Matrix")
            y_pred=model.predict(x_test)
            cm = confusion_matrix(y_test, y_pred, labels=model.classes_)
            ConfusionMatrixDisplay(cm,display_labels = class_names)
            st.pyplot()

        if 'ROC Curve' in metrics_list:
            st.subheader("ROC Curve")
            roc_curve(model, x_test, y_test)
            st.pyplot()

        if 'Precision-Recall Curve' in metrics_list:
            st.subheader("Precision-Recall Curve")
            precision_recall_curve(model, x_test, y_test)
            st.pyplot()

    df = load_data()
    x_train, x_test, y_train, y_test = split(df)
    class_names = ['edible', 'poisonous']
    st.sidebar.subheader("Choose Classifier")
    classifier = st.sidebar.selectbox("Classifier", ("Support Vector Machine (SVM)", "Logistic
Regression", "Random Forest"))

    if classifier == "Support Vector Machine (SVM)":
        st.sidebar.subheader("Model Hyperparameters")
        C = st.sidebar.number_input("C (Regularization parameter)", 0.01, 10.0, step = 0.01, key =
'C')
        kernel = st.sidebar.radio("Kernel", ("rbf", "linear"), key = 'kernel')
        gamma = st.sidebar.radio("Gamma (Kernel Coefficient)", ("scale", "auto"), key = 'auto')

        metrics = st.sidebar.multiselect("What metrics to plot?", ('Confusion Matrix', 'ROC Curve',
'Precision-Recall Curve'))

        if st.sidebar.button("Classify", key = 'classify'):
            st.subheader("Support Vector Machine (SVM) Results")
            model = SVC(C = C, kernel = kernel, gamma = gamma)
            model.fit(x_train, y_train)
            accuracy = model.score(x_test, y_test)
            y_pred = model.predict(x_test)
```

```
            st.write("Accuracy: ", accuracy)
            st.write("Precision: ", precision_score(y_test, y_pred, labels = class_names))
            st.write("Recall: ", recall_score(y_test, y_pred, labels = class_names))
            plot_metrics(metrics)

    if classifier == "Logistic Regression":
        st.sidebar.subheader("Model Hyperparameters")
        C = st.sidebar.number_input("C (Regularization parameter)", 0.01, 10.0, step = 0.01, key =
'C_LR')
        max_iter = st.sidebar.slider("Maximum number of iterations", 100, 500, key = 'max_iter')

        metrics = st.sidebar.multiselect("What metrics to plot?", ('Confusion Matrix', 'ROC Curve',
'Precision-Recall Curve'))

        if st.sidebar.button("Classify", key = 'classify'):
            st.subheader("Logistic Regression Results")
            model = LogisticRegression(C = C, max_iter = max_iter)
            model.fit(x_train, y_train)
            accuracy = model.score(x_test, y_test)
            y_pred = model.predict(x_test)
            st.write("Accuracy: ", accuracy)
            st.write("Precision: ", precision_score(y_test, y_pred, labels = class_names))
            st.write("Recall: ", recall_score(y_test, y_pred, labels = class_names))
            plot_metrics(metrics)

    if classifier == "Random Forest":
        st.sidebar.subheader("Model Hyperparameters")
        n_estimators = st.sidebar.number_input("The number of trees in the forest", 100, 5000,
step = 10, key = 'n_estimators')
        max_depth = st.sidebar.number_input("The maximum depth of the tree", 1, 20, step = 1,
key = 'max_depth')
        bootstrap = st.sidebar.radio("Bootstrap samples when building trees", ('True', 'False'), key =
'bootstrap')
        metrics = st.sidebar.multiselect("What metrics to plot?", ('Confusion Matrix', 'ROC Curve',
'Precision-Recall Curve'))

        if st.sidebar.button("Classify", key = 'classify'):
            st.subheader("Random Forest Results")
            model = RandomForestClassifier(n_estimators = n_estimators, max_depth =
max_depth, bootstrap = bootstrap, n_jobs = -1)
            model.fit(x_train, y_train)
            accuracy = model.score(x_test, y_test)
            y_pred = model.predict(x_test)
            st.write("Accuracy: ", accuracy)
```

```
            st.write("Precision: ", precision_score(y_test, y_pred, labels = class_names))
            st.write("Recall: ", recall_score(y_test, y_pred, labels = class_names))
            plot_metrics(metrics)

    if st.sidebar.checkbox("Show raw data", False):
        st.subheader("Mushroom Data Set (Classification)")
        st.write(df)

if __name__ == '__main__':
    main()
```