streamlit_ml_dashboard.py

```
import streamlit as st

from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report
from joblib import load

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scikitplot as skplt

from lime import lime_tabular

## Load Data
wine = load_wine()
wine_df = pd.DataFrame(wine.data, columns=wine.feature_names)
wine_df["WineType"] = [wine.target_names[typ] for typ in wine.target]

X_train, X_test, Y_train, Y_test = train_test_split(wine.data, wine.target,
train_size=0.8, random_state=123)

## Load Model
rf_classif = load("rf_classif.model")

Y_test_preds = rf_classif.predict(X_test)

## Dashboard
st.title("Wine Type :red[Prediction] :bar_chart: :chart_with_upwards_trend:
:tea: :coffee:")
st.markdown("Predict Wine Type using Ingredients Values")

tab1, tab2, tab3 = st.tabs(["Data :clipboard:", "Global Performance
:weight_lifter:", "Local Performance :bicyclist:"])

with tab1:
st.header("Wine Dataset")
st.write(wine_df)

with tab2:
st.header("Confusion Matrix | Feature Importances")
col1, col2 = st.columns(2)
with col1:
conf_mat_fig = plt.figure(figsize=(6,6))
ax1 = conf_mat_fig.add_subplot(111)
skplt.metrics.plot_confusion_matrix(Y_test, Y_test_preds, ax=ax1,
normalize=True)
st.pyplot(conf_mat_fig, use_container_width=True)
```

```
with col2:
feat_imp_fig = plt.figure(figsize=(6,6))
ax1 = feat_imp_fig.add_subplot(111)
skplt.estimators.plot_feature_importances(rf_classif,
feature_names=wine.feature_names, ax=ax1, x_tick_rotation=90)
st.pyplot(feat_imp_fig, use_container_width=True)

st.divider()
st.header("Classification Report")
st.code(classification_report(Y_test, Y_test_preds))

with tab3:
sliders = []
col1, col2 = st.columns(2)
with col1:
for ingredient in wine.feature_names:
ing_slider = st.slider(label=ingredient,
min_value=float(wine_df[ingredient].min()),
max_value=float(wine_df[ingredient].max()))
sliders.append(ing_slider)

with col2:
col1, col2 = st.columns(2, gap="medium")

prediction = rf_classif.predict([sliders])
with col1:
st.markdown("### Model Prediction :
{}".format(wine.target_names[prediction[0]]), unsafe_allow_html=True)

probs = rf_classif.predict_proba([sliders])
probability = probs[0][prediction[0]]

with col2:
st.metric(label="Model Confidence", value="{:.2f} %".format(probability*100),
delta="{:.2f} %".format((probability-0.5)*100))

explainer = lime_tabular.LimeTabularExplainer(X_train, mode="classification",
class_names=wine.target_names, feature_names=wine.feature_names)
explanation = explainer.explain_instance(np.array(sliders),
rf_classif.predict_proba, num_features=len(wine.feature_names), top_labels=3)
interpretation_fig = explanation.as_pyplot_figure(label=prediction[0])
st.pyplot(interpretation_fig, use_container_width=True)
```