```
# -*- coding: utf-8 -*-
"""Loan_Prediction (1).ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/15xZGK1P0WsU7TLd13id8Y8GUblw8LVJX
"""

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

test_data =
pd.read_csv("C:\\Users\\DSAI\\Downloads\\test_Y3wMUE5_7gLdaTN.csv")
train_data =
pd.read_csv("C:\\Users\\DSAI\\Downloads\\train_u6lujuX_CVtuZ9i.csv")

train_data.info()

test_data.info()

for column in train_data.columns:
    if train_data[column].dtype in ['float64', 'int64']:  # Numerical
columns
        mean_value = train_data[column].mean()
        train_data[column].fillna(mean_value, inplace=True)
    else:  # Categorical columns
        mode_value = train_data[column].mode()[0]
        train_data[column].fillna(mode_value, inplace=True)

for column in test_data.columns:
    if test_data[column].dtype in ['float64', 'int64']:  # Numerical
columns
        mean_value = test_data[column].mean()
        test_data[column].fillna(mean_value, inplace=True)
    else:  # Categorical columns
        mode_value = test_data[column].mode()[0]
        test_data[column].fillna(mode_value, inplace=True)

train_data.info()

test_data.info()

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in train_data.columns:
```

```
    train_data[col] = le.fit_transform(train_data[col])
for col in test_data.columns:
    test_data[col] = le.fit_transform(test_data[col])

!pip install seaborn

# Count plot for Loan Status
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='Loan_Status', data=train_data)
plt.title('Loan Status Distribution')
plt.show()

# Correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(train_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

train_data.drop(['Loan_ID'], axis=1, inplace=True)
test_data.drop(['Loan_ID'], axis=1, inplace=True)

# Histograms for numerical features
numerical_features = ['ApplicantIncome', 'CoapplicantIncome',
'LoanAmount', 'Loan_Amount_Term']
train_data[numerical_features].hist(bins=30, figsize=(10, 8))
plt.tight_layout()
plt.show()

# Boxplots for numerical features
for feature in numerical_features:
    plt.figure(figsize=(3, 2))
    sns.boxplot(y=train_data[feature])
    plt.title(f'Boxplot of {feature}')
    plt.show()

X = train_data.drop(columns=['Loan_Status'])
y = train_data['Loan_Status']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

#Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#model Building
#K-Nearest Neighbors (KNN)
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))

#Decision Trees
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

```
y_pred_dt = dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

#Support Vector Machine (SVM)
svm = SVC()
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))

#Logistic Regression (LR)
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))

#Naive Bayes (NB)
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))

#Random Forest (RF)
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))

#Comparing Model Performance
print("KNN Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
print("Decision Tree Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_dt))
print("SVM Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
print("Logistic Regression Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_lr))
print("Naive Bayes Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_nb))
print("Random Forest Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_rf))

# Print classification reports
print("KNN Classification Report:\n", classification_report(y_test,
y_pred_knn))
print("Decision Tree Classification Report:\n",
classification_report(y_test, y_pred_dt))
print("SVM Classification Report:\n", classification_report(y_test,
y_pred_svm))
print("Logistic Regression Classification Report:\n",
classification_report(y_test, y_pred_lr))
print("Naive Bayes Classification Report:\n",
classification_report(y_test, y_pred_nb))
print("Random Forest Classification Report:\n",
classification_report(y_test, y_pred_rf))

from sklearn.metrics import ConfusionMatrixDisplay

models = {
    'KNN': (knn, y_pred_knn),
    'Decision Tree': (dt, y_pred_dt),
```

```
    'SVM': (svm, y_pred_svm),
    'Logistic Regression': (lr, y_pred_lr),
    'Naive Bayes': (nb, y_pred_nb),
    'Random Forest': (rf, y_pred_rf)
}

for name, (model, y_pred) in models.items():
    cm = confusion_matrix(y_test, y_pred)
    # Check the number of classes in the confusion matrix (cm.shape[0])
    # and adjust display_labels accordingly
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=le.classes_[:cm.shape[0]])
    disp.plot(cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.show()

!pip install scikit-learn # install scikit-learn if it's not already
installed
from sklearn.metrics import roc_curve, auc # import the function
import matplotlib.pyplot as plt # import the plotting library

# Models that support predict_proba
prob_models = {
    'Logistic Regression': lr,
    'Naive Bayes': nb,
    'Random Forest': rf,
    'KNN': knn,
    'Decision Tree': dt
}

plt.figure(figsize=(10, 8))
for name, model in prob_models.items():
    y_proba = model.predict_proba(X_test)[:, 1]
    fpr, tpr, thresholds = roc_curve(y_test, y_proba)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.2f})')

# SVM requires special handling
if hasattr(svm, "decision_function"):
    y_score = svm.decision_function(X_test)
else:
    y_score = svm.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f'SVM (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
plt.show()

# Random Forest Feature Importance
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X.columns
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=feature_names[indices])
plt.title('Random Forest Feature Importances')
plt.show()

import matplotlib.pyplot as plt # import the plotting library
from sklearn.tree import plot_tree # import the plot_tree function

# Convert le.classes_ to a list of strings
class_names = [str(c) for c in le.classes_]

plt.figure(figsize=(20, 10))
plot_tree(dt, feature_names=X.columns, class_names=class_names,
filled=True) # Use the converted class names
plt.title('Decision Tree Visualization')
plt.show()

# Dictionary to store models and their predictions
model_predictions = {}

# List of models you have trained
models = {
    'KNN': knn,
    'Decision Tree': dt,
    'SVM': svm,
    'Logistic Regression': lr,
    'Naive Bayes': nb,
    'Random Forest': rf
}

# Make predictions on X_test_final for all models
for model_name, model in models.items():
    test_pred = model.predict(test_data)
    model_predictions[model_name] = test_pred  # Store predictions in
dictionary
    print(f"Predictions for {model_name}:\n", test_pred[:10])  # Print
first 10 predictions

pip install streamlit scikit-learn pandas joblib

import joblib

# Save models after training
joblib.dump(knn, 'knn_model.pkl')
joblib.dump(dt, 'dt_model.pkl')
joblib.dump(svm, 'svm_model.pkl')
joblib.dump(lr, 'lr_model.pkl')
joblib.dump(nb, 'nb_model.pkl')
joblib.dump(rf, 'rf_model.pkl')
```