

```
from google.colab import files
uploaded = files.upload()
```

[Choose Files](#) EasyVisa (1).csv

EasyVisa (1).csv(text/csv) - 1880839 bytes, last modified: 11/4/2025 - 100% done
Saving EasyVisa (1).csv to EasyVisa (1).csv

```
import os
os.rename("EasyVisa (1).csv", "EasyVisa.csv") # clean name
!ls
```

```
EasyVisa.csv  EasyVisa-ML-Classification  models  report  sample_data
```

```
import shutil
shutil.move("EasyVisa.csv", "/content/EasyVisa-ML-Classification/EasyVisa.csv")
!tree /content/EasyVisa-ML-Classification
```

```
/content/EasyVisa-ML-Classification
├── EasyVisa.csv
├── report
│   └── visuals
```

2 directories, 1 file

```
# --- setup folders ---
import os
os.makedirs("/content/EasyVisa-ML-Classification/report/visuals", exist_ok=True)

# --- load dataset (already moved in previous steps) ---
import pandas as pd
DATA_PATH = "/content/EasyVisa-ML-Classification/EasyVisa.csv"
df = pd.read_csv(DATA_PATH)
print("Loaded:", df.shape)
df.head()
```

Loaded: (25480, 12)

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_of_employees	yr_of_estab	regio
0	EZYV01	Asia	High School	N	N	14513	2007	
1	EZYV02	Asia	Master's	Y	N	2412	2002	
2	EZYV03	Asia	Bachelor's	N	Y	44444	2008	
3	EZYV04	Asia	Bachelor's	N	N	98	1897	
4	EZYV05	Africa	Master's	Y	N	1082	2005	

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# encode object columns
le = LabelEncoder()
df_encoded = df.copy()
for col in df_encoded.select_dtypes(include="object").columns:
    df_encoded[col] = le.fit_transform(df_encoded[col].astype(str))

# features/target
X = df_encoded.drop(columns=["case_status"])
y = df_encoded["case_status"]

# split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)
print("Train:", X_train.shape, " Test:", X_test.shape)
```

Train: (17836, 11) Test: (7644, 11)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```

clf = RandomForestClassifier(n_estimators=200, random_state=42)
clf.fit(X_train, y_train)
preds = clf.predict(X_test)

acc = accuracy_score(y_test, preds)
print(f"✅ Model trained. Accuracy: {acc:.4f}")
print(classification_report(y_test, preds))

```

```

✅ Model trained. Accuracy: 0.7877

```

	precision	recall	f1-score	support
0	0.80	0.91	0.85	5105
1	0.76	0.53	0.63	2539
accuracy			0.79	7644
macro avg	0.78	0.72	0.74	7644
weighted avg	0.78	0.79	0.78	7644

```

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, roc_curve, auc

# 4a) Confusion Matrix
cm = confusion_matrix(y_test, preds)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted"); plt.ylabel("Actual")
plt.tight_layout()
plt.savefig("/content/EasyVisa-ML-Classification/report/visuals/confusion_matrix.png", dpi=200)
plt.show()

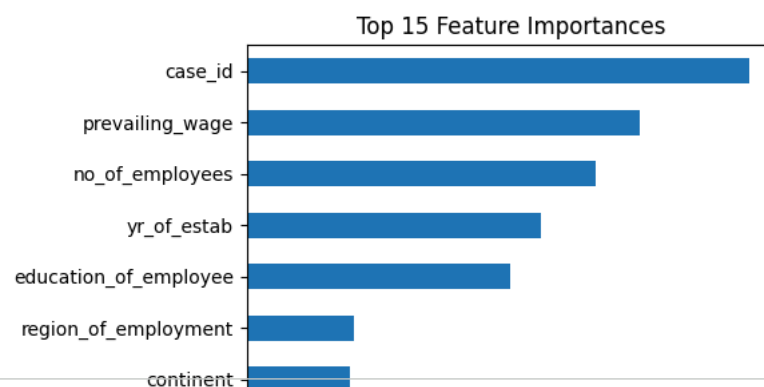
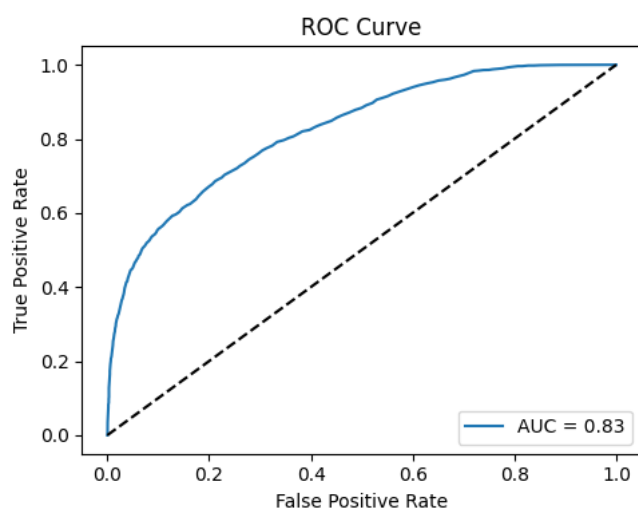
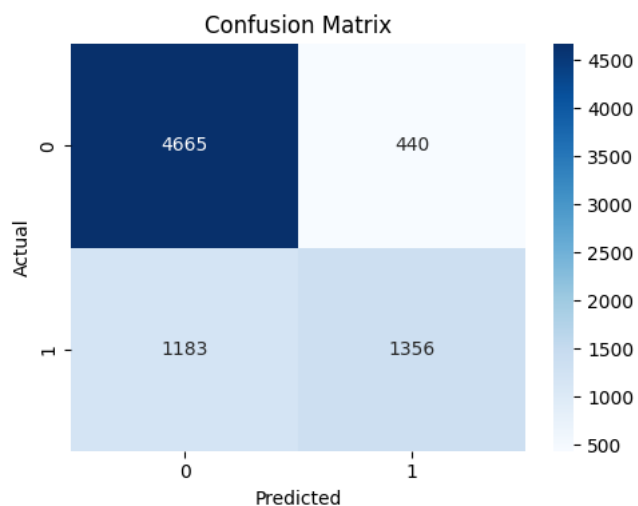
# 4b) ROC Curve (only if proba available)
if hasattr(clf, "predict_proba"):
    proba = clf.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, proba)
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(5,4))
    plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
    plt.plot([0,1],[0,1], 'k--')
    plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate")
    plt.title("ROC Curve"); plt.legend(loc="lower right")
    plt.tight_layout()
    plt.savefig("/content/EasyVisa-ML-Classification/report/visuals/roc_curve.png", dpi=200)
    plt.show()

# 4c) Feature Importance (tree-based)
import pandas as pd
import numpy as np
try:
    importances = clf.feature_importances_
    fi = pd.Series(importances, index=X_train.columns).sort_values(ascending=False).head(15)
    plt.figure(figsize=(6,5))
    fi[::-1].plot(kind="barh")
    plt.title("Top 15 Feature Importances")
    plt.tight_layout()
    plt.savefig("/content/EasyVisa-ML-Classification/report/visuals/feature_importance.png", dpi=200)
    plt.show()
except Exception as e:
    print("Feature importance not available:", e)

# 4d) Save classification report to CSV
rep = classification_report(y_test, preds, output_dict=True)
pd.DataFrame(rep).T.to_csv("/content/EasyVisa-ML-Classification/report/classification_report.csv", index=True)

print(f"✅ Saved: visuals + classification_report.csv")

```




```
import joblib, os
os.makedirs("/content/EasyVisa-ML-Classification/models", exist_ok=True)
joblib.dump(clf, "/content/EasyVisa-ML-Classification/models/best_model.joblib")
print("✅ Saved model to models/best_model.joblib")
```

✅ Saved model to models/best_model.joblib

```
import shutil, os
src = "/content/EasyVisa_CRISPDm_Report.pdf"
dst = "/content/EasyVisa-ML-Classification/report/EasyVisa_CRISPDm_Report.pdf"
if os.path.exists(src):
    shutil.move(src, dst)
!apt-get -qq install -y tree >/dev/null
!tree /content/EasyVisa-ML-Classification
```

```
/content/EasyVisa-ML-Classification
├── EasyVisa.csv
├── models
│   └── best_model.joblib
├── report
│   ├── classification_report.csv
│   └── visuals
│       └── confusion_matrix.png
```

 feature_importance.png
roc_curve.png

3 directories, 6 files