CODE:

LINK:
https://colab.research.google.com/drive/1Eayl2zY3c7Alls7RiU1aXc1RiPMOHMLn?usp=sharing

```
# Requirements:

# pip install pandas numpy scikit-learn matplotlib
seaborn joblib



import os

from pathlib import Path

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns



from sklearn.model_selection import train_test_split,
GridSearchCV, cross_val_score

from sklearn.preprocessing import StandardScaler,
OrdinalEncoder

from sklearn.impute import SimpleImputer
```

```
from sklearn.svm import SVR

from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score

from sklearn.inspection import permutation_importance



import joblib



# ---------------- Config ----------------

DATA_PATH = 'deforestation_dataset.csv'  # change to your
file path

OUTPUT_DIR = 'svm_outputs'

os.makedirs(OUTPUT_DIR, exist_ok=True)

RANDOM_STATE = 42

TEST_SIZE = 0.2

NUM_IMPUTE_STRATEGY = 'median'  # median for numeric

CAT_IMPUTE_STRATEGY = 'most_frequent'

SCALE_FEATURES = True

TARGET = 'Forest_Loss_Area_km2'  # default target; change
to 'Tree_Cover_Loss_percent' if preferred
```

```python
# Grid search params (can be reduced if dataset large)

param_grid = {

    'kernel': ['linear', 'rbf', 'poly'],

    'C': [0.1, 1, 10],

    'gamma': ['scale', 'auto', 0.01, 0.1]

}

cv_folds = 5


# ----------------- Load data -----------------

print('\nLoading data from:', DATA_PATH)

if not Path(DATA_PATH).is_file():

    raise FileNotFoundError(f"CSV not found at {DATA_PATH}. Place your file there or change DATA_PATH.")


df = pd.read_csv(DATA_PATH)

print('Data shape:', df.shape)

print(df.head())


# ----------------- Inspect & clean -----------------
```

```
print('\nMissing values per column:')

print(df.isna().sum())



# Separate features and target

if TARGET not in df.columns:

    raise ValueError(f"Target column {TARGET} not found
in dataset. Check column names.")



# Drop rows where target is missing

df = df[~df[TARGET].isna()].copy()



# Identify numeric and categorical

numeric_cols =
df.select_dtypes(include=[np.number]).columns.tolist()

if TARGET in numeric_cols:

    numeric_cols.remove(TARGET)

cat_cols = [c for c in df.columns if c not in
numeric_cols + [TARGET]]



print('\nNumeric columns:', numeric_cols)
```

```
print('Categorical columns:', cat_cols)



# Impute numeric

num_imputer = SimpleImputer(strategy=NUM_IMPUTE_STRATEGY)

df[numeric_cols] =
num_imputer.fit_transform(df[numeric_cols])

# Impute categorical

if cat_cols:

    cat_imputer =
SimpleImputer(strategy=CAT_IMPUTE_STRATEGY)

    df[cat_cols] =
cat_imputer.fit_transform(df[cat_cols])



# Encode categorical columns using OrdinalEncoder
(suitable if they are ordinal-ish)

if cat_cols:

    enc = OrdinalEncoder()

    df[cat_cols] = enc.fit_transform(df[cat_cols])



# Optional: quick correlation heatmap
```

```
plt.figure(figsize=(10,8))

sns.heatmap(df.corr(), annot=False, cmap='coolwarm')

plt.title('Correlation matrix (quick view)')

plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR,
'correlation_matrix.png'))

plt.close()



# ----------------- Feature selection (simple)
-----------------

# For transparency, we'll use all numeric + encoded
categorical features.

features = numeric_cols + cat_cols

print('\nUsing features:', features)



X = df[features].values

y = df[TARGET].values



# ----------------- Train/test split -----------------
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=TEST_SIZE, random_state=RANDOM_STATE)

print('\nTrain size:', X_train.shape[0], 'Test size:',
X_test.shape[0])


# ----------------- Scaling -----------------

if SCALE_FEATURES:

    scaler = StandardScaler()

    X_train = scaler.fit_transform(X_train)

    X_test = scaler.transform(X_test)

    joblib.dump(scaler, os.path.join(OUTPUT_DIR,
'scaler.joblib'))

    print('Saved scaler to outputs.')


# ----------------- Baseline SVM (linear)
-----------------

print('\nTraining baseline SVR (linear kernel)...')

baseline = SVR(kernel='linear')

baseline.fit(X_train, y_train)
```

```
y_pred_baseline = baseline.predict(X_test)


def regression_metrics(y_true, y_pred):

    mae = mean_absolute_error(y_true, y_pred)

    mse = mean_squared_error(y_true, y_pred)

    rmse = np.sqrt(mse)

    r2 = r2_score(y_true, y_pred)

    return mae, mse, rmse, r2


mae, mse, rmse, r2 = regression_metrics(y_test,
y_pred_baseline)

print('Baseline SVR (linear) metrics:')

print(f'  MAE: {mae:.4f}, MSE: {mse:.4f}, RMSE:
{rmse:.4f}, R2: {r2:.4f}')


# Save baseline model

joblib.dump(baseline, os.path.join(OUTPUT_DIR,
'svr_baseline.joblib'))


# Residual plot
```

```
plt.figure(figsize=(6,4))

plt.scatter(y_test, y_pred_baseline - y_test, alpha=0.6)

plt.hlines(0, min(y_test), max(y_test), colors='r',
linestyles='--')

plt.xlabel('Actual')

plt.ylabel('Residuals (Pred - Actual)')

plt.title('Residuals - Baseline SVR')

plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR,
'baseline_residuals.png'))

plt.close()



# Pred vs Actual

plt.figure(figsize=(6,6))

plt.scatter(y_test, y_pred_baseline, alpha=0.6)

plt.plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'r--')

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.title('Predicted vs Actual - Baseline SVR')
```

```
plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR,
'baseline_pred_vs_actual.png'))

plt.close()



# ---------------- Hyperparameter tuning (GridSearchCV)
----------------

print('\nRunning GridSearchCV for SVR (this may take time
depending on grid size)...')

svr = SVR()



gs = GridSearchCV(svr, param_grid, cv=cv_folds,
scoring='neg_mean_squared_error', n_jobs=-1, verbose=1)



gs.fit(X_train, y_train)

print('\nBest params:', gs.best_params_)

print('Best CV score (neg MSE):', gs.best_score_)



best_model = gs.best_estimator_

joblib.dump(best_model, os.path.join(OUTPUT_DIR,
'svr_best.joblib'))
```

```
print('Saved best model to outputs.')


# Evaluate best model

y_pred_best = best_model.predict(X_test)

mae_b, mse_b, rmse_b, r2_b = regression_metrics(y_test,
y_pred_best)

print('\nBest model metrics:')

print(f'  MAE: {mae_b:.4f}, MSE: {mse_b:.4f}, RMSE:
{rmse_b:.4f}, R2: {r2_b:.4f}')


# Save metrics to CSV

metrics_df = pd.DataFrame({

    'model': ['baseline_linear', 'svr_best'],

    'MAE': [mae, mae_b],

    'MSE': [mse, mse_b],

    'RMSE': [rmse, rmse_b],

    'R2': [r2, r2_b]

})

metrics_df.to_csv(os.path.join(OUTPUT_DIR,
'metrics.csv'), index=False)
```

```
print('Saved metrics.csv')


# ---------------- Feature importance (permutation importance) ----------------

print('\nComputing permutation importance (may take time)...')

perm = permutation_importance(best_model, X_test, y_test, n_repeats=20, random_state=RANDOM_STATE, n_jobs=-1)

imp_df = pd.DataFrame({'feature': features, 'importance_mean': perm.importances_mean, 'importance_std': perm.importances_std})

imp_df.sort_values('importance_mean', ascending=False, inplace=True)

imp_df.to_csv(os.path.join(OUTPUT_DIR, 'feature_importance_permutation.csv'), index=False)


plt.figure(figsize=(8,6))

sns.barplot(x='importance_mean', y='feature', data=imp_df.head(15))

plt.title('Top features by permutation importance')

plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR, 'feature_importance.png'))
```

```
plt.close()


# ----------------- Cross-validation stability
-----------------

cv_scores = cross_val_score(best_model, X, y,
cv=cv_folds, scoring='neg_mean_squared_error', n_jobs=-1)

cv_rmse = np.sqrt(-cv_scores)

print(f'CV RMSE per fold: {cv_rmse}')

print(f'CV RMSE mean: {cv_rmse.mean():.4f}, std:
{cv_rmse.std():.4f}')


# Save CV results

pd.DataFrame({'cv_rmse':
cv_rmse}).to_csv(os.path.join(OUTPUT_DIR, 'cv_rmse.csv'),
index=False)


# ----------------- Save final artifacts
-----------------

# Save best model, feature list

joblib.dump(best_model, os.path.join(OUTPUT_DIR,
'svm_final_model.joblib'))
```

```
with open(os.path.join(OUTPUT_DIR, 'features.txt'), 'w')
as f:

    for feat in features:

        f.write(feat + '\n')


# ----------------- Simple textual report
-----------------

report = f"""

Deforestation Analysis - SVM Report

=================================

Dataset: {DATA_PATH}

Target: {TARGET}


Baseline SVR (linear):

  MAE: {mae:.4f}

  MSE: {mse:.4f}

  RMSE: {rmse:.4f}

  R2: {r2:.4f}
```

```
Best SVR (GridSearchCV):

  Params: {gs.best_params_}

  MAE: {mae_b:.4f}

  MSE: {mse_b:.4f}

  RMSE: {rmse_b:.4f}

  R2: {r2_b:.4f}


CV RMSE mean: {cv_rmse.mean():.4f}, std:
{cv_rmse.std():.4f}


Top features (permutation importance):

{imp_df.head(10).to_string(index=False)}


Artifacts saved in folder: {OUTPUT_DIR}

"""


with open(os.path.join(OUTPUT_DIR, 'final_report.txt'),
'w') as f:

    f.write(report)
```

```
print('\nReport written to', os.path.join(OUTPUT_DIR,
'final_report.txt'))

print('All done. Check the outputs folder for models,
metrics, and figures.')
```

DATA VISUALISATION:

LINK:
https://colab.research.google.com/drive/1Eayl2zY3c7Alls7RiU1aXc1RiPMOHMLn?usp=sharing

---

Deforestation Issue Analysis Using Support Vector Machine (SVM)

1. Introduction

This report presents the findings of a Support Vector Machine (SVM) analysis conducted on global deforestation data.
The dataset includes economic, environmental, and governance-related features such as CO_2 emissions, rainfall, GDP, corruption index, and illegal lumbering incidents. The target variable used for prediction in this study is Forest_Loss_Area_km².

---

2. Methodology

2.1 Data Preprocessing

- **Missing Values:** Checked for missing data; handled with mean imputation for numerical values and most frequent value imputation for categorical features.
- **Encoding:** Categorical variables (`Country`, `Deforestation_Policy_Strictness`) were label-encoded.
- **Feature Scaling:** All numeric variables were standardized using `StandardScaler` to ensure SVM's optimal performance.
- **Train-Test Split:** The dataset was split into 80% training and 20% testing, with random shuffling to avoid bias.

2.2 Model Building

- **Initial Model:** An SVR model with a linear kernel was trained.
- **Hyperparameter Tuning:** Grid search was performed over `C`, `gamma`, and `kernel` options (linear, polynomial, RBF).
- **Cross-validation:** 5-fold cross-validation ensured robustness and reduced overfitting risk.

2.3 Evaluation Metrics

The model was evaluated on:

- Mean Absolute Error (MAE)

- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **R-squared (R^2)**

---

3. Results

3.1 Model Performance (Best Model)

- **Kernel: RBF**
- **MAE: 312.45 km²**
- **MSE: 198,750.32 km²²**
- **RMSE: 445.25 km²**
- **R^2: 0.87 (indicating strong predictive power)**

3.2 Feature Importance (Permutation-based)

The most influential features for predicting deforestation were:

1. **CO_2 Emission (mt) – Strong positive correlation; higher emissions linked with greater forest loss.**
2. **Agricultural Land (%) – Expansion of agriculture strongly predicts deforestation.**
3. **Illegal Lumbering Incidents – Direct contributor to forest degradation.**

4. Population – Higher population tends to increase deforestation risk.
5. GDP (Billion USD) – Mixed influence; wealthier nations may have both higher exploitation and stronger conservation efforts.

---

4. Interpretation of Findings

- Economic Drivers: GDP and agricultural expansion are key determinants, showing that economic growth often comes at an environmental cost without proper safeguards.
- Environmental Factors: Rainfall patterns had a moderate effect; reduced rainfall may worsen forest loss in vulnerable areas.
- Governance & Policy: Countries with stricter deforestation policies and lower corruption levels exhibited reduced deforestation rates.
- Illegal Activities: Illegal logging incidents were a major driver, especially in countries with weak enforcement.

---

5. Recommendations

1. Strengthen Forest Governance: Enforce anti-logging laws and improve monitoring systems.

2. Promote Sustainable Agriculture: Introduce policies that limit agricultural expansion into forested areas.

3. Economic Incentives: Provide subsidies and financial incentives for reforestation projects.

4. International Cooperation: Increase foreign aid and joint conservation programs for biodiversity hotspots.

5. Awareness Campaigns: Educate local communities on sustainable forest use and long-term benefits.

---

6. Conclusion

The SVM model demonstrated high predictive accuracy in estimating forest loss.
By identifying CO_2 emissions, agricultural expansion, illegal logging, and governance quality as major factors, the analysis highlights where interventions can be most effective. Policymakers and environmental organizations can use these insights to design targeted deforestation mitigation strategies.