

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import KNNImputer
from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import
accuracy_score, precision_score, recall_score, f1_score, classification_report
```

```
df = pd.read_csv('/content/drive/MyDrive/Projects/Project2/moral_machine_responses.csv')
df.head()
```

```
# Because dataset is too large
df = df.sample(50000, random_state=42)
```

```
# Check missing values
df.isnull().sum()
```

```
# Fill missing values
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0]
    df[column] = df[column].fillna(mode_value)
```

```
X=df.drop(['saved','outcome_id'],axis=1)
y=df['saved']
```

```
le=LabelEncoder()
for c in X.select_dtypes(include='object').columns:
    X[c]=le.fit_transform(X[c])
```

```
numeric_cols=X.select_dtypes(include='number').columns
imputer=KNNImputer(n_neighbors=2)
X[numeric_cols]=imputer.fit_transform(X[numeric_cols])
```

```

scaler=StandardScaler()
X=scaler.fit_transform(X)

X_train,X_test,y_train,y_test=train_test_split(
    X,y,test_size=0.25,random_state=42)

models={
    'Logistic Regression':(
        LogisticRegression(max_iter=1000),
        {'C':[0.1,1,10]}
    ),
    'Decision Tree':(
        DecisionTreeClassifier(random_state=42),
        {'max_depth':[5,10,None]}
    ),
    'Random Forest':(
        RandomForestClassifier(random_state=42),
        {'n_estimators':[100,200]}
    ),
    'SVM':(
        SVC(),
        {'C':[0.1,1], 'kernel':['linear','rbf']}
    ),
    'KNN':(
        KNeighborsClassifier(),
        {'n_neighbors':[3,5,7]}
    ),
    'Naive Bayes':(
        GaussianNB(),
        {}
    )
}

results=[]

for name,(model,params) in models.items():
    if params:
        grid=GridSearchCV(model,params,cv=3,n_jobs=-1)
        grid.fit(X_train,y_train)
        best=grid.best_estimator_
    else:
        best=model.fit(X_train,y_train)

    pred=best.predict(X_test)

    results.append([
        name,
        accuracy_score(y_test,pred),
        precision_score(y_test,pred),
        recall_score(y_test,pred),

```

```

        f1_score(y_test,pred)
    ])

pd.DataFrame(results,columns=['Model','Accuracy','Precision','Recall','F1'])

# Feature Selection - Chi Square

X_df=pd.DataFrame(X)

selector=SelectKBest(score_func=chi2,k=10)
X_chi=selector.fit_transform(X_df-X_df.min().min(),y)

X_train,X_test,y_train,y_test=train_test_split(
    X_chi,y,test_size=0.25,random_state=42)

rf=RandomForestClassifier(random_state=42)
rf.fit(X_train,y_train)
pred=rf.predict(X_test)

print(classification_report(y_test,pred))

# Feature Selection - RFE

X_df=pd.DataFrame(X)

rfe=RFE(LogisticRegression(max_iter=1000),n_features_to_select=10)
X_rfe=rfe.fit_transform(X_df,y)

X_train,X_test,y_train,y_test=train_test_split(
    X_rfe,y,test_size=0.25,random_state=42)

rf=RandomForestClassifier(random_state=42)
rf.fit(X_train,y_train)
pred=rf.predict(X_test)

print(classification_report(y_test,pred))

```