```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier

fema = pd.read_csv('/content/final_project_FEMA (1).csv')
len(fema)
```

```
4457134
```

```
sum_tsaEligible = fema['tsaEligible'].sum()
length_fema = len(fema)
result = sum_tsaEligible / length_fema
print(result)
```

```
0.3857961192102369
```

```
fema = fema.dropna()
len(fema)
fema = fema.drop(['damagedStateAbbreviation','damagedCity','disasterNumber'],axis = 1)


X = fema.drop(['tsaEligible'], axis = 1)
Y = fema['tsaEligible']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)


preprocessor = ColumnTransformer(
    transformers=[
        ('num', MinMaxScaler(), ['waterLevel', 'grossIncome', 'householdComposition', 'repairAmount']),
        ('cat', OneHotEncoder(handle_unknown='ignore'), ['residenceType'])
    ], remainder ='passthrough')

pipeline = Pipeline(steps=[('preprocessor', preprocessor)])

X_train = pipeline.fit_transform(X_train)

X_test = pipeline.transform(X_test)


#rf = RandomForestClassifier(n_estimators=200, max_depth= None , min_samples_split=10, min_samples_leaf=2)
dt = DecisionTreeClassifier(max_depth=10, min_samples_leaf = 2, criterion ='entropy', min_samples_split = 10, random_state=42)
dt.fit(X_train, Y_train)
Y_pred = dt.predict(X_test)

dt_cm = confusion_matrix(Y_test, Y_pred)
print("Confusion Matrix:")
print(dt_cm)

dt_accuracy = accuracy_score(Y_test, Y_pred)
print(f"Accuracy: {dt_accuracy}")

dt_precision = precision_score(Y_test, Y_pred)
print(f"Precision: {dt_precision}")

#precision results were inconsistent, was always above .66, most common seen was .668
```

```
Confusion Matrix:
[[510780  36779]
 [272266  71602]]
Accuracy: 0.6533142927014776
Precision: 0.6606508520866203
```

```
## depending on dataset will get different results, most consistent was 122k eligible.
new = pd.read_csv('/content/new_disaster (1).csv')
```

```
new_damaged_city = new[['damagedCity']]


new = new.drop(['damagedCity','disasterNumber'], axis = 1)


new_processed = pipeline.transform(new)
new_pred = dt.predict(new_processed)
new_pred
```

```
array([0, 1, 0, ..., 0, 0, 0])
```

```
new['tsa_eligible'] = new_pred
new.to_csv('new_predictions.csv', index=False)


tsa_eligible_count = new[new['tsa_eligible'] == 1].shape[0]
print(f"Number of TSA eligible people in the new dataset: {tsa_eligible_count}")
```

```
Number of TSA eligible people in the new dataset: 108874
```

```
len(new)
```

```
898761
```

```
tsa_eligible_count/len(new)
```

```
0.121137877589259
```

```
new.head()
```

	Property_Number	specialNeeds	roofDamage	foundationDamage	householdComposition	destroyed	residenceType	repairAmount	grossIncom
0	1	0.0	0.0	0.0	2.0	0.0	Apartment	0.00	52000.(
1	2	0.0	1.0	0.0	4.0	0.0	House/Duplex	3698.52	30000.(
2	3	0.0	0.0	0.0	2.0	0.0	House/Duplex	0.00	3780.(
3	4	0.0	0.0	0.0	3.0	0.0	House/Duplex	0.00	18048.(
4	5	0.0	0.0	0.0	2.0	0.0	House/Duplex	0.00	0.(

```
household_composition_greater_than_2 = new[new['householdComposition'] > 2]
count = household_composition_greater_than_2[household_composition_greater_than_2['tsa_eligible'] == 1].shape[0]
print(f"Number of household compositions greater than 2 with tsa_eligible of 1: {count}")
```

```
Number of household compositions greater than 2 with tsa_eligible of 1: 44081
```

```
household_composition_greater_than_2 = new[new['householdComposition'] > 2]
count = household_composition_greater_than_2[household_composition_greater_than_2['tsa_eligible'] == 0].shape[0]
print(f"Number of household compositions greater than 2 with tsa_eligible of 0: {count}")
```

```
Number of household compositions greater than 2 with tsa_eligible of 0: 262918
```

```
for i in range(len(new_pred)):
  if new_pred[i] == 1:
      feature_importances = dt.feature_importances_
      most_important_feature_index = feature_importances.argmax()
      feature_names = list(pipeline.named_steps['preprocessor'].get_feature_names_out())
      print(f"For the prediction at index {i} (tsa_eligible = 1), the most important feature was {feature_names[most_important_feature_index
      break
```

```
For the prediction at index 1 (tsa_eligible = 1), the most important feature was num__repairAmount with an importance of 0.2987539915253
```