Fraud-detection-with-machine-learning

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
```

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

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/AI Files/paymentfraud.csv")
df.head()
```

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
0	29	1	4.745402	paypal	28.204861	0
1	725	1	4.742303	storecredit	0.000000	0
2	845	1	4.921318	creditcard	0.000000	0
3	503	1	4.886641	creditcard	0.000000	0
4	2000	1	5.040929	creditcard	0.000000	0

Next steps: Generate code with df View recommended plots

```
# Split dataset up into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df.drop('label', axis=1), df['label'],
    test_size=0.33, random_state=17)
```

```
# Assuming 'label' is the column containing string labels
X = df.drop('label', axis=1)
y = df['label']
```

```
# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Create a column transformer with one-hot encoding for categorical columns
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_columns)
    ],
    remainder='passthrough'
)
```

```
# Create a pipeline with the preprocessor and the logistic regression model
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=17)
```

```
# Fit the model
clf.fit(X_train, y_train)
```

```
Pipeline
preprocessor: ColumnTransformer
cat
OneHotEncoder
remainder
passthrough
LogisticRegression
```

```
# Make predictions on the test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 1.0
```

```
# Compare test set predictions with ground truth labels
print(confusion_matrix(y_test, y_pred))
```

```
[[12753     0]
 [    0   190]]
```

Using RandomForestClassifier

```
# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns
```

```
# Create a column transformer with one-hot encoding for categorical columns
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_columns)
    ],
    remainder='passthrough'
)
```

```
# Create a pipeline with the preprocessor and the Random Forest classifier
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier())
])

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=27)

# Fit the model
clf.fit(X_train, y_train)
```

```
Pipeline
preprocessor: ColumnTransformer
cat
OneHotEncoder
remainder
passthrough
RandomForestClassifier
```

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9722629993046434
```

```
# Make predictions on the test set
y_pred = clf.predict(X_test)

# Compute and print confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[12768     0]
 [    0   175]]
```