In [ ]:

```
import pandas as pd
import numpy as np

import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Question 1. What Percentage of the users have churned in the data provided?

In [ ]:

```
#Please refer to code signal submission for detailed explaination
    # read files
    features_data = pd.read_csv('../input/dataset0108/features_data.csv')
    equity_value_data = pd.read_csv('../input/dataset0108/equity_value_data.csv')

    churned_user_list = []
    current_user = equity_value_data['user_id'][0]
    count = 0
    for index in range(len(equity_value_data['user_id'])-1):
        if equity_value_data['user_id'][index] != current_user:
            if count >= 28:
                churned_user_list.append(current_user)
            count = 0
            current_user = equity_value_data['user_id'][index]
        else:
            if equity_value_data['close_equity'][index] < 100:
                count += 1
            else:
                if count >= 28:
                    churned_user_list.append(current_user)
                count = 0

    churned_header_list = []

    for i in range(len(features_data['user_id'])):

        if features_data['user_id'][i] in churned_user_list:
            churned_header_list.append('yes')
        else:
            churned_header_list.append('no')
    #print(chunked_header_list)

    features_data['churned'] = churned_header_list
```

In [ ]:

```
features_data.head(5)
```

In [ ]:

```
#calculation for percentage finding
churned_yes = features_data.churned.value_counts().yes
churned_no = features_data.churned.value_counts().no
```

In [ ]:

```
percentage_churned = (churned_yes / (churned_yes + churned_no)) * 100
```

Solution is below i.e. the percentage

In [ ]:

```
print( 'Percentage of churned users is',percentage_churned)
```

Question 2

In [ ]:

```
df = features_data
```

In [ ]:

```
features_data.head(5)
```

In [ ]:

```
# converting the categorical Values into integers for classification
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['risk_tolerance']=le.fit_transform(df['risk_tolerance'].astype(str))
df['investment_experience']=le.fit_transform(df['investment_experience'].astype(str))
df['liquidity_needs']=le.fit_transform(df['liquidity_needs'].astype(str))
df['platform']=le.fit_transform(df['platform'].astype(str))
df['instrument_type_first_traded']=le.fit_transform(df['risk_tolerance'].astype(int))
df['time_horizon']=le.fit_transform(df['investment_experience'].astype(str))
df['churned']=le.fit_transform(df['churned'].astype(str))
```

Question - What are the top 3 features that have high correlation with Churn

In [ ]:

```
#plotting matrix to find High correlation
import seaborn as sns
corrmat = features_data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
g = sns.heatmap(df[top_corr_features].corr(),annot=True,cmap='RdYlGn')
```

Clearly, investment_experience, instrument_type_first_traded and risk_tolerance are the most important features that are highly correlated with the user Churn. These are used in the prediction of the user churn.

Question - What is the distribution of feature that has highest correlation with churn

In [ ]:

```
#solution - investment_experience has a highest correlation with churn which is left skew
ed as shown in the histogram
df['investment_experience'].hist()
```

solution - investment_experience has a highest correlation with churn which is left skewed as shown in the histogram

Question - Build Model to Predict Churn probability and Find AUC on the Test data

In [ ]:

```
# function to find out the AUC scores for multiple models
from sklearn.model_selection import RepeatedStratifiedKFold
def get_scores(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X_test, y_test, scoring='roc_auc', cv=cv, n_jobs=-1)

    return scores
```

In [ ]:

```
del df['user_id']
```

In [ ]:

```
df.head(2)
```

In [ ]:

```
X=df
X
```

In [ ]:

```
y = df['churned']
y
```

In [ ]:

```
del X['churned']
```

In [ ]:

```
# K fold cross validation to handle class imbalance problem and to reduce overfitting
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model_selection import KFold

scores = []
#best_svr = SVR(kernel='rbf')
cv = KFold(n_splits=2, random_state=42, shuffle=True)
for train_index, test_index in cv.split(df):
    print("Train Index: ", train_index, "\n")
    print("Test Index: ", test_index)

    X_train, X_test, y_train, y_test = X.iloc[train_index], X.iloc[test_index], y.iloc[t
rain_index], y.iloc[test_index]
    #best_svr.fit(X_train, y_train)
    #scores.append(best_svr.score(X_test, y_test))

    print("AUC Scores using Decision Tree Classifier is ",get_scores(DecisionTreeClassifi
er(),X_train, X_test, y_train, y_test))
    print("")
    print("AUC Scores using Random Forest Ensemble method is",get_scores(RandomForestCla
ssifier(),X_train, X_test, y_train, y_test))
    print("")
    print("AUC Scores using Support Vector Machine is",get_scores(SVC(),X_train, X_test,
y_train, y_test))
    print("")
    #print(get_score(SVC(),X_train, X_test, y_train, y_test))
    #print(get_score(RandomForestClassifier(),X_train, X_test, y_train, y_test))
```

In [ ]:

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```