Structured Data Assignment

Documentation

Problem 1 - The development of drugs is critical in providing therapeutic options for patients suffering from chronic and terminal illnesses. “Target Drug”, in particular, is designed to enhance the patient's health and well-being without causing dependence on other medications that could potentially lead to severe and life-threatening side effects. These drugs are specifically tailored to treat a particular disease or condition, offering a more focused and effective approach to treatment, while minimising the risk of harmful reactions. The objective in this assignment is to develop a predictive model which will predict whether a patient will be eligible for “Target Drug” or not in next 30 days. Knowing if the patient is eligible or not will help physician treating the patient make informed decision on which treatments to give. A patient is considered eligible for a particular drug when they have taken their first prescription for that drug.

The objective is to develop a predictive model to determine whether a patient will be eligible for "Target Drug" in the next 30 days. This prediction is important for physicians to make informed treatment decisions. A patient is considered eligible for "Target Drug" when they have taken their first prescription for that drug. The dataset contains information about patients, their prescription history, and incidents related to drug treatments.

Steps:

Data Preprocessing

1. Import necessary libraries for data analysis and modeling, including pandas, numpy, seaborn, matplotlib, datetime, and scikit-learn modules.
2. Load the training data from a Parquet file using pd.read_parquet.
3. Display the first few rows of the training data using train_data.head().
4. Generate summary statistics of the training data using train_data.describe().
5. Check for missing values in the training data using train_data.isnull().sum().
6. Identify and count duplicated rows in the training data using train_data.duplicated().sum().
7. Remove duplicate rows from the training data using train_data = train_data.drop_duplicates().
8. Check for duplicated rows after removal using train_data.duplicated().sum().
9. Display the data types of each column in the training data using train_data.dtypes.
10. Output unique values of the 'Incident' column using print(train_data['Incident'].unique()).

Data Preparation

1. Filter the training data to obtain records with 'Incident' as 'TARGET DRUG' to create a dataset of positive cases.
2. Create a subset of training data containing records not related to 'TARGET DRUG' to create a dataset of negative cases.
3. Select the most recent record for each patient in the negative dataset to avoid duplication.
4. Perform cumulative counting of prescriptions for 'TARGET DRUG' for both positive and negative datasets to create 'Prescription_Count' columns.
5. Calculate the time difference (days) between the prediction date (30 days from today) and the last prescription date for both positive and negative datasets to create 'Time_diff' columns.

Model Building and Evaluation

1. Concatenate the positive and negative data to create the final dataset.
2. Split the final dataset into training and testing sets using train_test_split.
3. Build a predictive model, train it on the training set, and predict the target variable on the test set.
4. Calculate the accuracy score of the model using accuracy_score.
5. Calculate the false positive rate, true positive rate, and threshold for the ROC curve.
6. Calculate the area under the ROC curve (AUC).

7. Plot the ROC curve and display the AUC value.

Test Data Preprocessing

1. Load the test data from a Parquet file.
2. Display the first few and last few rows of the test data.
3. Generate summary statistics of the test data.
4. Check for missing values in the test data.
5. Identify and count duplicated rows in the test data.
6. Remove duplicate rows from the test data.
7. Output unique values of the 'Incident' column in the test data.
8. Create a dataset of positive cases in the test data similar to the training data.
9. Create a dataset of negative cases in the test data.
10. Select the most recent record for each patient in the negative dataset.
11. Perform cumulative counting of prescriptions and calculate the time difference for both positive and negative cases.

Creating a Final Submission

1. Concatenate the positive and negative data in the test data to create a new dataset.
2. Drop any duplicate rows in the training data using train_data.drop_duplicates(inplace=True).
3. Calculate the prescription count and time difference for the test data based on the prediction date.
4. Save the final submission as a CSV file named 'final_submission.csv' using final_submission.to_csv('final_submission.csv', index=False).

Problem Statement Summary

The development of "Target Drug" is essential for treating chronic and terminal illnesses. The predictive model aims to determine a patient's eligibility for "Target Drug" in the next 30 days. This prediction is vital for informed treatment decisions by physicians. A patient is considered eligible when they take their first prescription for "Target Drug." The project demonstrates data pre-processing, model building, evaluation, and creating a final submission.

```
# Import necessary libraries for data manipulation and analysis
import pandas as pd

# Import NumPy for numerical computations
import numpy as np

# Import Seaborn for data visualization
import seaborn as sns

# Import Matplotlib for creating plots and graphs
import matplotlib.pyplot as plt

# Import datetime for handling date and time-related operations
from datetime import datetime, timedelta

# Import functions and classes for machine learning tasks
from sklearn.model_selection import train_test_split, cross_val_score, cross_validate
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score, roc_auc_score as ras, roc_curve, auc, accuracy_score


# Import the Logistic Regression model for classification
from sklearn.linear_model import LogisticRegression


# Import the Random Forest classifier for ensemble learning
from sklearn.ensemble import RandomForestClassifier


# Read the training data from a Parquet file into a Pandas DataFrame
train_data = pd.read_parquet("sample_data/train.parquet")


# Display the first few rows of the training data
train_data.head()
```

	Patient-Uid	Date	Incident
0	a0db1e73-1c7c-11ec-ae39-16262ee38c7f	2019-03-09	PRIMARY_DIAGNOSIS
1	a0dc93f2-1c7c-11ec-9cd2-16262ee38c7f	2015-05-16	PRIMARY_DIAGNOSIS
3	a0dc94c6-1c7c-11ec-a3a0-16262ee38c7f	2018-01-30	SYMPTOM_TYPE_0
4	a0dc950b-1c7c-11ec-b6ec-16262ee38c7f	2015-04-22	DRUG_TYPE_0
8	a0dc9543-1c7c-11ec-bb63-16262ee38c7f	2016-06-18	DRUG_TYPE_1

```
# Generate summary statistics for the training data
train_data.describe()
```

```
<ipython-input-8-7e87d3225b90>:2: FutureWarning: Treating datetime data as categorica
  train_data.describe()
```

	Patient-Uid	Date	Incident
count	3220868	3220868	3220868
unique	27033	1977	57
top	a0ddfd2c-1c7c-11ec-876d-16262ee38c7f	2019-05-21 00:00:00	DRUG_TYPE_6
freq	1645	3678	561934
first	NaN	2015-04-07 00:00:00	NaN
last	NaN	2020-09-03 00:00:00	NaN

```
# Check for missing values in the training data
train_data.isnull().sum()
```

```
Patient-Uid    0
Date           0
Incident       0
dtype: int64
```

```
# Check for duplicate rows and remove them from the training data
train_data.duplicated().sum()
```

```
35571
```

```
# Remove duplicates from the training data
train_data = train_data.drop_duplicates()
```

```
# Check for duplicate rows after removing them
train_data.duplicated().sum()
```

```
0
```

```
# Display the data types of the columns in the training data
train_data.dtypes
```

```
Patient-Uid            object
Date           datetime64[ns]
Incident               object
dtype: object
```

```
# Extract unique values of the 'Incident' column in the training data
print("Unique values of Incident \n")
print(train_data['Incident'].unique())
```

```
Unique values of Incident

['PRIMARY_DIAGNOSIS' 'SYMPTOM_TYPE_0' 'DRUG_TYPE_0' 'DRUG_TYPE_1'
 'DRUG_TYPE_2' 'TEST_TYPE_0' 'DRUG_TYPE_3' 'DRUG_TYPE_4' 'DRUG_TYPE_5'
 'DRUG_TYPE_6' 'DRUG_TYPE_8' 'DRUG_TYPE_7' 'SYMPTOM_TYPE_1' 'DRUG_TYPE_10'
 'SYMPTOM_TYPE_29' 'SYMPTOM_TYPE_2' 'DRUG_TYPE_11' 'DRUG_TYPE_9'
 'DRUG_TYPE_13' 'SYMPTOM_TYPE_5' 'TEST_TYPE_1' 'SYMPTOM_TYPE_6'
 'TEST_TYPE_2' 'SYMPTOM_TYPE_3' 'SYMPTOM_TYPE_8' 'DRUG_TYPE_14'
 'DRUG_TYPE_12' 'SYMPTOM_TYPE_9' 'SYMPTOM_TYPE_10' 'SYMPTOM_TYPE_7'
 'SYMPTOM_TYPE_11' 'TEST_TYPE_3' 'DRUG_TYPE_15' 'SYMPTOM_TYPE_4'
 'SYMPTOM_TYPE_14' 'SYMPTOM_TYPE_13' 'SYMPTOM_TYPE_16' 'SYMPTOM_TYPE_17'
 'SYMPTOM_TYPE_15' 'SYMPTOM_TYPE_18' 'SYMPTOM_TYPE_12' 'SYMPTOM_TYPE_20'
 'SYMPTOM_TYPE_21' 'DRUG_TYPE_17' 'SYMPTOM_TYPE_22' 'TEST_TYPE_4'
 'SYMPTOM_TYPE_23' 'DRUG_TYPE_16' 'TEST_TYPE_5' 'SYMPTOM_TYPE_19'
 'SYMPTOM_TYPE_24' 'SYMPTOM_TYPE_25' 'SYMPTOM_TYPE_26' 'SYMPTOM_TYPE_27'
 'DRUG_TYPE_18' 'SYMPTOM_TYPE_28' 'TARGET DRUG']
```

```
# Filter the positive data with 'Incident' equal to 'TARGET DRUG'
positive_data = train_data[train_data['Incident']=='TARGET DRUG']
positive_data.head()
```

	Patient-Uid	Date	Incident
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET DRUG
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET DRUG
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET DRUG
3309423	a0ecc615-1c7c-11ec-aa31-16262ee38c7f	2019-11-15	TARGET DRUG
3309494	a0ea612f-1c7c-11ec-8cf0-16262ee38c7f	2020-03-18	TARGET DRUG

```
# Create a negative dataset by excluding patients from the positive dataset
negative = train_data[~train_data['Patient-Uid'].isin(positive_data['Patient-Uid'])]
negative_data = negative.groupby('Patient-Uid').tail(1)
negative_data
```

	Patient-Uid	Date	Incident
1560892	a0e3a8c0-1c7c-11ec-98c2-16262ee38c7f	2018-05-06	PRIMARY_DIAGNOSIS
1620903	a0dd6a3f-1c7c-11ec-9b86-16262ee38c7f	2015-04-07	SYMPTOM_TYPE_0
1629044	a0e48a75-1c7c-11ec-8c5f-16262ee38c7f	2018-08-22	DRUG_TYPE_6
1942882	a0e3cf61-1c7c-11ec-8098-16262ee38c7f	2018-08-21	DRUG_TYPE_2
1975541	a0e91a8c-1c7c-11ec-acc2-16262ee38c7f	2020-04-15	PRIMARY_DIAGNOSIS
...	...	...	...
3256795	a0e045a1-1c7c-11ec-8014-16262ee38c7f	2020-07-10	PRIMARY_DIAGNOSIS
3256799	a0e67e2a-1c7c-11ec-b805-16262ee38c7f	2015-12-16	PRIMARY_DIAGNOSIS
3256800	a0dec400-1c7c-11ec-80df-16262ee38c7f	2019-08-06	PRIMARY_DIAGNOSIS
3256804	a0e09919-1c7c-11ec-9e7d-16262ee38c7f	2017-02-19	DRUG_TYPE_6
3256805	a0e69331-1c7c-11ec-a98d-16262ee38c7f	2015-10-03	DRUG_TYPE_6

17659 rows × 3 columns

```
# Calculate the 'Prescription_Count' for both positive and negative datasets
positive_data['Prescription_Count'] = positive_data.groupby('Patient-Uid')['Date'].cumcount()
negative_data['Prescription_Count'] = negative_data.groupby('Patient-Uid')['Date'].cumcount()
positive_data.tail(5)
```

```
<ipython-input-17-1a72b642929e>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  positive_data['Prescription_Count'] = positive_data.groupby('Patient-Uid')['Date'].
<ipython-input-17-1a72b642929e>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  negative_data['Prescription_Count'] = negative_data.groupby('Patient-Uid')['Date'].
```

	Patient-Uid	Date	Incident	Prescription_Count
29074998	a0ef2b6d-1c7c-11ec-9172-16262ee38c7f	2018-10-12	TARGET DRUG	4
29075105	a0ebe423-1c7c-11ec-a5e0-16262ee38c7f	2019-07-02	TARGET DRUG	9
29075494	a0ebc713-1c7c-11ec-bd53-16262ee38c7f	2019-05-21	TARGET DRUG	10

```
negative_data.tail()
```

	Patient-Uid	Date	Incident	Prescription_Count
3256795	a0e045a1-1c7c-11ec-8014-16262ee38c7f	2020-07-10	PRIMARY_DIAGNOSIS	0
3256799	a0e67e2a-1c7c-11ec-b805-16262ee38c7f	2015-12-16	PRIMARY_DIAGNOSIS	0
3256800	a0dec400-1c7c-11ec-80df-16262ee38c7f	2019-08-06	PRIMARY_DIAGNOSIS	0
	a0e09919-1c7c-11ec-	2017-		

```
# Calculate the 'Time_diff' for both datasets based on a prediction date
prediction_date = pd.to_datetime('today') + pd.DateOffset(days=30)
positive_data['Time_diff'] = (prediction_date - positive_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
negative_data['Time_diff'] = (prediction_date - negative_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
```

```
<ipython-input-19-cdfc937a0060>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  positive_data['Time_diff'] = (prediction_date - positive_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
<ipython-input-19-cdfc937a0060>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  negative_data['Time_diff'] = (prediction_date - negative_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
```

```
positive_data.head()
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET DRUG	0	1197
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET DRUG	0	1451
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET DRUG	0	1476
	a0ecc615-1c7c-11ec-	2019-	TARGET		

```
negative_data.head()
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
	a0e3a8c0-				

```
# Concatenate the positive and negative datasets to create the final dataset
final_data = pd.concat([positive_data, negative_data])
final_data.head()
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET DRUG	0	1197
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET DRUG	0	1451
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET DRUG	0	1476
	a0ecc615-1c7c-11ec-	2019-	TARGET		

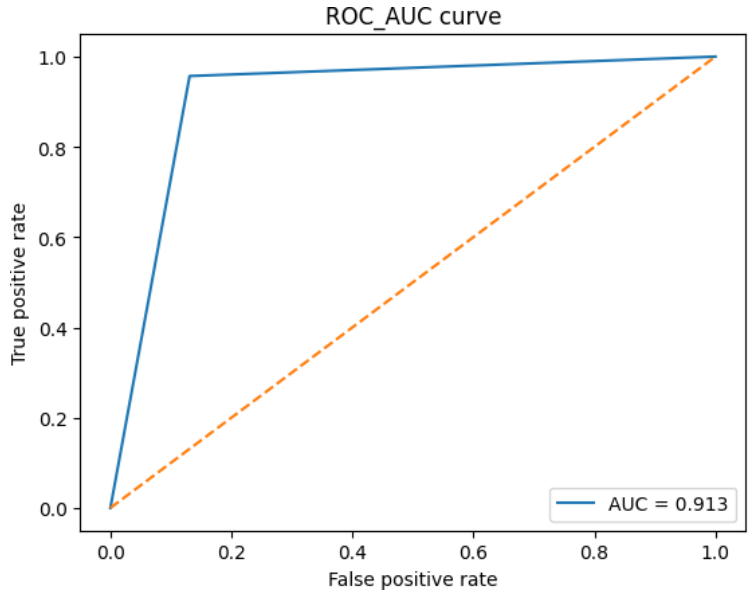
```
# Split the final dataset into training and testing sets
X_train,X_test,y_train,y_test = X_train, X_test, y_train, y_test = train_test_split(final_data[['Prescription_Count', 'Time_diff']], fina


# Train a machine learning model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model using the F1-score
y_pred = model.predict(X_test)
f1 = f1_score(y_test, y_pred)
print("F1-score:", f1)
```

```
F1-score: 0.961200978929147
```

```
# Calculate and display the accuracy score of the model on the test set
accuracy_score(y_test, y_pred)
```

```
0.9387370405278039
```

```
# Calculate and plot the ROC-AUC curve
fpr,tpr, thresold = roc_curve(y_test, y_pred)
roc_auc = auc(fpr,tpr)
plt.plot(fpr,tpr, label = 'AUC = %0.3f' % roc_auc)
plt.plot([0,1],[0,1],'--')
plt.title('ROC_AUC curve')
plt.legend(loc='lower right')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.show()
```




```
# Load the test data from a Parquet file
test_data = pd.read_parquet("sample_data/test.parquet")


# Display the first 5 rows of the test data
test_data.head()
```

	Patient-Uid	Date	Incident
0	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2016-12-08	SYMPTOM_TYPE_0
1	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2018-10-17	DRUG_TYPE_0
2	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2017-12-01	DRUG_TYPE_2
3	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2018-12-05	DRUG_TYPE_1
4	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2017-11-04	SYMPTOM_TYPE_0

```
# Display the last 5 rows of the test data
test_data.tail()
```

	Patient-Uid	Date	Incident
1372854	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2017-05-11	DRUG_TYPE_13
1372856	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2018-08-22	DRUG_TYPE_2
1372857	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2017-02-04	DRUG_TYPE_2
1372858	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2017-09-25	DRUG_TYPE_8
1372859	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2017-05-19	DRUG_TYPE_7

```
# Generate descriptive statistics for the test data
test_data.describe()
```

```
<ipython-input-30-0777917ce040>:1: FutureWarning: Treating datetime data as categoric
  test_data.describe()
```

	Patient-Uid	Date	Incident
count	1065524	1065524	1065524
unique	11482	1947	55
top	a0faa6ed-1c7c-11ec-8f6f-16262ee38c7f	2018-03-13 00:00:00	DRUG_TYPE_6
freq	1236	1139	192292
first	NaN	2015-04-07 00:00:00	NaN
last	NaN	2020-08-04 00:00:00	NaN

```
# Check for missing (null) values in the test data
test_data.isnull().sum()
```

```
Patient-Uid    0
Date           0
Incident       0
dtype: int64
```

```
# Calculate the number of duplicated rows in the test data
test_data.duplicated().sum()
```

```
12100
```

```
# Remove duplicate rows from the test data
test_data = test_data.drop_duplicates()
```

```
# Recalculate the number of duplicated rows after removing duplicates
test_data.duplicated().sum()
```

```
0
```

```
# Print unique values in the 'Incident' column of the test data
print("Unique values of Incident \n")
print(test_data['Incident'].unique())
```

```
Unique values of Incident

['SYMPTOM_TYPE_0' 'DRUG_TYPE_0' 'DRUG_TYPE_2' 'DRUG_TYPE_1'
 'PRIMARY_DIAGNOSIS' 'DRUG_TYPE_8' 'TEST_TYPE_0' 'DRUG_TYPE_7'
 'DRUG_TYPE_11' 'SYMPTOM_TYPE_6' 'DRUG_TYPE_5' 'DRUG_TYPE_6' 'DRUG_TYPE_9'
 'DRUG_TYPE_15' 'TEST_TYPE_3' 'SYMPTOM_TYPE_3' 'TEST_TYPE_1' 'DRUG_TYPE_3'
 'TEST_TYPE_2' 'SYMPTOM_TYPE_7' 'DRUG_TYPE_12' 'SYMPTOM_TYPE_2'
 'SYMPTOM_TYPE_10' 'SYMPTOM_TYPE_1' 'SYMPTOM_TYPE_17' 'SYMPTOM_TYPE_18'
 'SYMPTOM_TYPE_5' 'SYMPTOM_TYPE_15' 'SYMPTOM_TYPE_9' 'SYMPTOM_TYPE_4'
 'SYMPTOM_TYPE_8' 'SYMPTOM_TYPE_29' 'DRUG_TYPE_13' 'SYMPTOM_TYPE_21'
```

```
'DRUG_TYPE_4' 'SYMPTOM_TYPE_12' 'SYMPTOM_TYPE_11' 'SYMPTOM_TYPE_19'
'DRUG_TYPE_14' 'SYMPTOM_TYPE_16' 'TEST_TYPE_4' 'DRUG_TYPE_10'
'SYMPTOM_TYPE_26' 'SYMPTOM_TYPE_14' 'SYMPTOM_TYPE_24' 'DRUG_TYPE_16'
'SYMPTOM_TYPE_13' 'TEST_TYPE_5' 'SYMPTOM_TYPE_20' 'SYMPTOM_TYPE_25'
'SYMPTOM_TYPE_22' 'DRUG_TYPE_17' 'SYMPTOM_TYPE_27' 'SYMPTOM_TYPE_23'
'SYMPTOM_TYPE_28']
```

```
# Filter the test data to create a subset of records with 'Incident' equal to 'TARGET DRUG'
positive_data = test_data[test_data['Incident']=='TARGET DRUG']
# Display the first 5 rows of the positive data
positive_data.head()
```

	Patient-Uid	Date	Incident

```
# Create a subset of records in the test data where 'Incident' is not 'TARGET DRUG'
negative =  test_data[~test_data['Patient-Uid'].isin(positive_data['Patient-Uid'])]
# Group the negative data by 'Patient-Uid' and select the last record for each group
negative_data = negative.groupby('Patient-Uid').tail(1)
# Display the negative data
negative_data
```

	Patient-Uid	Date	Incident
57	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2017-12-01	TEST_TYPE_0
208	a0f9e9f9-1c7c-11ec-b565-16262ee38c7f	2016-06-22	DRUG_TYPE_9
305	a0f9ea43-1c7c-11ec-aa10-16262ee38c7f	2019-07-21	DRUG_TYPE_6
420	a0f9ea7c-1c7c-11ec-af15-16262ee38c7f	2016-06-15	DRUG_TYPE_6
497	a0f9eab1-1c7c-11ec-a732-16262ee38c7f	2018-11-22	DRUG_TYPE_6
...	...	...	...
1372381	a102720c-1c7c-11ec-bd9a-16262ee38c7f	2020-01-07	DRUG_TYPE_6
1372432	a102723c-1c7c-11ec-9f80-16262ee38c7f	2019-07-06	DRUG_TYPE_3
1372543	a102726b-1c7c-11ec-bfbf-16262ee38c7f	2018-12-31	DRUG_TYPE_0
1372607	a102729b-1c7c-11ec-86ba-16262ee38c7f	2019-04-02	DRUG_TYPE_3
1372859	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2017-05-19	DRUG_TYPE_7

11482 rows × 3 columns

```
# Calculate the prescription count for each record in the positive data based on 'Date'
positive_data['Prescription_Count'] = positive_data.groupby('Patient-Uid')['Date'].cumcount()
# Calculate the prescription count for each record in the negative data based on 'Date'
negative_data['Prescription_Count'] = negative_data.groupby('Patient-Uid')['Date'].cumcount()
# Display the last 5 rows of the positive data
positive_data.tail(5)
```

```
<ipython-input-38-1a72b642929e>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  negative_data['Prescription_Count'] = negative_data.groupby('Patient-Uid')['Date'].
```

	Patient-Uid	Date	Incident	Prescription_Count

```
# Display the last 5 rows of the negative data
negative_data.tail()
```

	Patient-Uid	Date	Incident	Prescription_Count
1372381	a102720c-1c7c-11ec-bd9a-16262ee38c7f	2020-01-07	DRUG_TYPE_6	0
1372432	a102723c-1c7c-11ec-9f80-16262ee38c7f	2019-07-06	DRUG_TYPE_3	0
1372543	a102726b-1c7c-11ec-bfbf-16262ee38c7f	2018-12-31	DRUG_TYPE_0	0
	a102729b-1c7c-11ec-86ba-	2019-04-		

```
# Calculate the prediction date as today's date plus 30 days
prediction_date = pd.to_datetime('today') + pd.DateOffset(days=30)
# Calculate the time difference for each record in the positive dat
positive_data['Time_diff'] = (prediction_date - positive_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
# Calculate the time difference for each record in the negative data
negative_data['Time_diff'] = (prediction_date - negative_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
```

```
<ipython-input-40-cdfc937a0060>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  negative_data['Time_diff'] = (prediction_date - negative_data.groupby('Patient-Uid')['Date'].transform('max')).dt.days
```

```
# Display the first 5 rows of the positive data
positive_data.head()
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff

```
# Display the first 5 rows of the negative data
negative_data.head()
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
57	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2017-12-01	TEST_TYPE_0	0	2183
208	a0f9e9f9-1c7c-11ec-b565-16262ee38c7f	2016-06-22	DRUG_TYPE_9	0	2710
305	a0f9ea43-1c7c-11ec-aa10-16262ee38c7f	2019-07-21	DRUG_TYPE_6	0	1586
	a0f9ea7c-1c7c-11ec-	2016-			

```
# Concatenate the positive and negative data to create a new data frame
new_data = pd.concat([positive_data, negative_data])
# Display the first 5 rows of the new data
new_data.head()
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
57	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	2017-12-01	TEST_TYPE_0	0	2183
208	a0f9e9f9-1c7c-11ec-b565-16262ee38c7f	2016-06-22	DRUG_TYPE_9	0	2710
305	a0f9ea43-1c7c-11ec-aa10-16262ee38c7f	2019-07-21	DRUG_TYPE_6	0	1586
	a0f9ea7c-1c7c-11ec-	2016-			

```
# Display the first 5 rows of the new data
train_data.drop_duplicates(inplace = True)


# Calculate the prescription count for each record in the test data based on 'Date'
test_data['Prescription_Count'] = test_data.groupby('Patient-Uid')['Date'].cumcount()
# Calculate the time difference for each record in the test data
test_data['Time_Difference'] = (prediction_date - test_data.groupby('Patient-Uid')['Date'].transform(max)).dt.days


# Create a RandomForestClassifier instance
clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model on your training data
clf.fit(X_train, y_train)

# Make predictions on the test data
test_data_pred = clf.predict(X_test)


test_data_pred
```

```
array([ True,  True,  True, ...,  True,  True, False])
```

```
# Filter the 'Patient-Uid' column to match the length of the predictions
patient_uid_subset = test_data['Patient-Uid'].iloc[:len(test_data_pred)]

# Create the DataFrame
final_submission = pd.DataFrame({'Patient-Uid': patient_uid_subset, 'Prediction': test_data_pred})
final_submission.head()
```

	Patient-Uid	Prediction
0	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	True
1	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	True
2	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	True
3	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	True
4	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	True

```
Save the final_submission DataFrame to a CSV file named 'final_submission.csv' without including the index column.
final_submission.to_csv('final_submission.csv', index = False)
```