

Topic: Predicting LogP and TPSA of ChEMBL compounds

Okay, I understand. You're working with ChEMBL 35 data, using RDKit, and following an AIMLops-inspired directory structure. You're running into issues with your SQL query (regex operator incompatibility) and scikit-learn version. You need assistance in structuring your code within the AIMLops framework, generating SQL and Python code snippets, and addressing the errors you're encountering. And you want to limit the data to 100 rows for faster prototyping.

Here's a breakdown, analysis, code examples, and solutions, formatted for your use case, and presented in both English and Vietnamese.

1. Analysis of the Analysis Model:

Your analysis model likely involves extracting bioactivity data from the ChEMBL database, filtering it based on specific criteria (e.g., activity type, standard value), and then using this data in Python with RDKit for tasks such as:

- **Structure-Activity Relationship (SAR) Modeling:** Building models to predict activity based on molecular features.
- **Virtual Screening:** Identifying potential drug candidates by screening a library of compounds against a target.
- **Lead Optimization:** Improving the properties of existing drug candidates.
- **Data Exploration and Visualization:** Understanding trends in the data through plots and statistical analysis.

The SQL code serves to extract and prepare the data. The Python code uses RDKit for molecular manipulation and potentially scikit-learn for machine learning models.

Vietnamese Translation:

Mô hình phân tích của bạn có khả năng liên quan đến việc trích xuất dữ liệu hoạt tính sinh học từ cơ sở dữ liệu ChEMBL, lọc dữ liệu dựa trên các tiêu chí cụ thể (ví dụ: loại hoạt động, giá trị tiêu chuẩn), sau đó sử dụng dữ liệu này trong Python với RDKit cho các tác vụ như:

- **Mô hình hóa mối quan hệ cấu trúc-hoạt tính (SAR):** Xây dựng mô hình để dự đoán hoạt động dựa trên các đặc điểm phân tử.
- **Sàng lọc ảo:** Xác định các ứng cử viên thuốc tiềm năng bằng cách sàng lọc một thư viện các hợp chất đối với một mục tiêu.
- **Tối ưu hóa dẫn đầu:** Cải thiện các thuộc tính của các ứng cử viên thuốc hiện có.
- **Khám phá và trực quan hóa dữ liệu:** Hiểu các xu hướng trong dữ liệu thông qua các biểu đồ và phân tích thống kê.

Mã SQL phục vụ để trích xuất và chuẩn bị dữ liệu. Mã Python sử dụng RDKit để thao tác phân tử và có khả năng sử dụng scikit-learn cho các mô hình học máy.

2. AIMLops Directory Structure and Code Integration

Assuming a simplified AIMLops-inspired structure, it might look like this:

```
Topic_ChEMBL_35_11/  
├── data/  
│   └── chembl_activity_data.csv  
└── notebooks/
```

```

├── Topic_CheMBL_35_11_1_data_extraction_and_cleaning.ipynb
├── Topic_CheMBL_35_11_2_sar_analysis.ipynb
├── src/
│   └── utils.py # Optional: Reusable functions
└── README.md

```

- **data/**: Stores your extracted data (e.g., chembl_activity_data.csv).
- **notebooks/**: Contains your Jupyter notebooks.
- **src/**: (Optional) Holds Python modules with reusable functions.

Vietnamese Translation:

Giả sử một cấu trúc lấy cảm hứng từ AIMLops đơn giản, nó có thể trông như thế này:

```

Topic_CheMBL_35_11/
├── data/
│   └── chembl_activity_data.csv
├── notebooks/
│   ├── Topic_CheMBL_35_11_1_data_extraction_and_cleaning.ipynb
│   └── Topic_CheMBL_35_11_2_phan_tich_sar.ipynb
├── src/
│   └── utils.py # Tùy chọn: Các hàm có thể tái sử dụng
└── README.md

```

- **data/**: Lưu trữ dữ liệu đã trích xuất của bạn (ví dụ: chembl_activity_data.csv).
- **notebooks/**: Chứa các sổ tay Jupyter của bạn.
- **src/**: (Tùy chọn) Chứa các mô-đun Python với các hàm có thể tái sử dụng.

3. SQL Code (Addressing Regex Issue and Limiting to 100 Rows)

The error “ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ ‘+ \$’” indicates that the ~ operator (PostgreSQL’s regular expression match) is not compatible with the numeric data type in the standard_value column. We need to cast the column to text *before* applying the regex. Also, the LIMIT 100 clause will restrict the output to 100 rows.

```

-- File: data/chembl_activity_data.sql
SELECT
    act.activity_id,
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    mol.molfile
FROM activities act
JOIN assays ass ON act.assay_id = ass.assay_id
JOIN target_dictionary td ON ass.tid = td.tid
JOIN molecule_dictionary md ON act.molregno = md.molregno
JOIN compound_structures cmp ON md.molregno = cmp.molregno
LEFT JOIN mols mol ON md.molregno = mol.molregno
WHERE td.target_type = 'SINGLE PROTEIN'
    AND ass.assay_type = 'B'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value::TEXT ~ '^[0-9.]+$' -- Cast to TEXT before regex
LIMIT 100;

```

Explanation:

¹ 0-9.

- `act.standard_value::TEXT ~ '^[0-9.]+$'` : This line explicitly casts the `standard_value` column to the TEXT data type before applying the regular expression. `^[0-9.]+$` ensures that the value consists only of digits and periods.
- `LIMIT 100`: Restricts the number of rows returned to 100.

How to run:

1. Open pgAdmin and connect to your `chembl_35` database.
2. Open a new query window.
3. Paste the SQL code above into the query window.
4. Execute the query.
5. **Important:** In pgAdmin, use the “Copy with Headers” option after running the query to copy the result to your clipboard. Then, save this clipboard content as a CSV file named `chembl_activity_data.csv` inside your `data/` directory. Alternatively, explore pgAdmin’s export functionality for more robust CSV creation.

Vietnamese Translation:

- `act.standard_value::TEXT ~ '^[0-9.]+$'` : Dòng này chuyển đổi rõ ràng cột `standard_value` thành kiểu dữ liệu TEXT trước khi áp dụng biểu thức chính quy. `^[0-9.]+$` đảm bảo rằng giá trị chỉ bao gồm các chữ số và dấu chấm.
- `LIMIT 100`: Hạn chế số lượng hàng trả về là 100.

Hướng dẫn chạy:

1. Mở pgAdmin và kết nối với cơ sở dữ liệu `chembl_35` của bạn.
2. Mở một cửa sổ truy vấn mới.
3. Dán mã SQL ở trên vào cửa sổ truy vấn.
4. Thực thi truy vấn.
5. **Quan trọng:** Trong pgAdmin, sử dụng tùy chọn “Sao chép với tiêu đề” sau khi chạy truy vấn để sao chép kết quả vào clipboard của bạn. Sau đó, lưu nội dung clipboard này dưới dạng tệp CSV có tên `chembl_activity_data.csv` bên trong thư mục `data/` của bạn. Ngoài ra, hãy khám phá chức năng xuất của pgAdmin để tạo CSV mạnh mẽ hơn.

4. Python Code (Example Notebooks)

Here are two example notebooks to demonstrate how to read the data, use RDKit, and perform some basic analysis.

Notebook 1: `notebooks/Topic_CheMBL_35_11_1_data_extraction_and_cleaning.ipynb`

```
# ---
# jupyter:
#   jupyter:
#     text_representation:
#       extension: .py
#       format_name: percent
#       format_version: '1.3'
#       jupyter_text_version: 1.14.5
#     kernelspec:
#       display_name: Python 3 (ipykernel)
#       language: python
#       name: python3
# ---

# %% [markdown]
# # Topic_CheMBL_35_11_1: Data Extraction and Cleaning

# %%
```

```

import pandas as pd
import os
from rdkit import Chem

# Define the base path
base_path = os.path.dirname(os.getcwd())
data_path = os.path.join(base_path, 'data', 'chembl_activity_data.csv')

# %%
# Load the data
try:
    df = pd.read_csv(data_path)
except FileNotFoundError:
    print(f"Error: File not found at {data_path}. Make sure the CSV file exists.")
    exit()

# %%
# Display the first few rows
print(df.head())

# %%
# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

# %%
# Basic data cleaning: Drop rows with missing 'molfile'
df = df.dropna(subset=['molfile'])

# %%
# Convert molfile to RDKit Mol object
def molfile_to_mol(molfile):
    try:
        mol = Chem.MolFromMolBlock(molfile)
        if mol is None:
            return None
        return mol
    except:
        return None

df['mol'] = df['molfile'].apply(molfile_to_mol)

# Drop rows where the conversion to Mol object failed
df = df.dropna(subset=['mol'])

# Display the cleaned data
print("\nCleaned Data:")
print(df.head())

# %%
# Basic data cleaning: keep the IC50 data type.
df = df[df['standard_type'] == 'IC50']

# %%
# Save the cleaned data (optional)
cleaned_data_path = os.path.join(base_path, 'data',
'chembl_activity_data_cleaned.csv')
df.to_csv(cleaned_data_path, index=False)
print(f"\nCleaned data saved to {cleaned_data_path}")

```

Notebook 2: notebooks/Topic_CheMBL_35_11_2_sar_analysis.ipynb

```
# ---
# jupyter:
# jupyter:
#   text_representation:
#     extension: .py
#     format_name: percent
#     format_version: '1.3'
#     jupyter_text_version: 1.14.5
#   kernelspec:
#     display_name: Python 3 (ipykernel)
#     language: python
#     name: python3
# ---

# %% [markdown]
# # Topic_CheMBL_35_11_2: SAR Analysis

# %%
import pandas as pd
import os
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Define the base path
base_path = os.path.dirname(os.getcwd())
cleaned_data_path = os.path.join(base_path, 'data',
'chembl_activity_data_cleaned.csv')

# %%
# Load the cleaned data
try:
    df = pd.read_csv(cleaned_data_path)
except FileNotFoundError:
    print(f"Error: File not found at {cleaned_data_path}. Make sure you run the first notebook first.")
    exit()

# Convert molfile to RDKit Mol object
def molfile_to_mol(molfile):
    try:
        mol = Chem.MolFromMolBlock(molfile)
        if mol is None:
            return None
        return mol
    except:
        return None

df['mol'] = df['molfile'].apply(molfile_to_mol)

# Drop rows where the conversion to Mol object failed
df = df.dropna(subset=['mol'])

# %%
```

```

# Calculate molecular weight
def calculate_mw(mol):
    return Descriptors.MolWt(mol)

df['mol_weight'] = df['mol'].apply(calculate_mw)

# %%
# Prepare data for modeling
X = df[['mol_weight']] # Feature: Molecular Weight
y = -np.log10(df['standard_value']) # Target: pIC50 (transformed IC50)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# %%
# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# %%
# Make predictions
y_pred = model.predict(X_test)

# %%
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# %%
# Plot predictions vs. actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual pIC50")
plt.ylabel("Predicted pIC50")
plt.title("Actual vs. Predicted pIC50")
plt.show()

```

Explanation:

- **os.path.join(base_path, ...):** This ensures your code works regardless of where the notebook is run, as long as the directory structure is maintained.
- **RDKit Chem.MolFromMolBlock(molfile):** This converts the molfile string (representing the molecular structure) into an RDKit Mol object, which can then be used for calculations.
- **Descriptors.MolWt(mol):** This calculates the molecular weight of the molecule.
- **-np.log10(df['standard_value']):** This transforms the IC50 value to pIC50 (a logarithmic scale), which is often a better representation for SAR modeling.
- **train_test_split, LinearRegression, mean_squared_error, r2_score:** These are scikit-learn functions for splitting data, training a linear regression model, and evaluating its performance. The squared=False parameter is *not* needed for mean_squared_error in current versions of scikit-learn.

Vietnamese Translation:

- **os.path.join(base_path, ...):** Điều này đảm bảo mã của bạn hoạt động bất kể sổ tay được chạy ở đâu, miễn là cấu trúc thư mục được duy trì.

- `RDKit.Chem.MolFromMolBlock(molfile)`: Điều này chuyển đổi chuỗi molfile (đại diện cho cấu trúc phân tử) thành đối tượng Mol của RDKit, sau đó có thể được sử dụng cho các phép tính.
- `Descriptors.MolWt(mol)`: Điều này tính toán trọng lượng phân tử của phân tử.
- `-np.log10(df['standard_value'])`: Điều này chuyển đổi giá trị IC50 thành pIC50 (thang logarit), thường là một biểu diễn tốt hơn cho mô hình hóa SAR.
- `train_test_split, LinearRegression, mean_squared_error, r2_score`: Đây là các hàm scikit-learn để chia dữ liệu, huấn luyện mô hình hồi quy tuyến tính và đánh giá hiệu suất của nó. Tham số `squared=False` không cần thiết cho `mean_squared_error` trong các phiên bản scikit-learn hiện tại.

5. Addressing Errors

- **ERROR: operator does not exist: numeric ~ unknown**: Add `::TEXT` to cast the numeric column to text before applying the regex (as shown in the SQL code above).
- **old scikit-learn version does not support parameters `squared=False` in the `mean_squared_error` function**: Remove the `squared=False` parameter from the `mean_squared_error` function. The default behavior is to return the mean squared error. If you need the root mean squared error, calculate it manually: `rmse = np.sqrt(mean_squared_error(y_test, y_pred))`

Vietnamese Translation:

- *ERROR: operator does not exist: numeric ~ unknown*: Thêm `::TEXT` để chuyển đổi cột số thành văn bản trước khi áp dụng regex (như được hiển thị trong mã SQL ở trên).
- *old scikit-learn version does not support parameters `squared=False` in the `mean_squared_error` function*: Xóa tham số `squared=False` khỏi hàm `mean_squared_error`. Hành vi mặc định là trả về lỗi bình phương trung bình. Nếu bạn cần căn bậc hai của lỗi bình phương trung bình, hãy tính nó thủ công: `rmse = np.sqrt(mean_squared_error(y_test, y_pred))`

6. Five Code Examples (Extending the Analysis)

Here are five additional code examples to extend your analysis, focusing on different aspects of the data and RDKit:

Example 1: Calculating LogP

```
# In notebooks/Topic_CheMBL_35_11_2_sar_analysis.ipynb (or a new notebook)
from rdkit.Chem import AllChem
from rdkit.Chem import Crippen

def calculate_logp(mol):
    return Crippen.MolLogP(mol)

df['logp'] = df['mol'].apply(calculate_logp)
print(df[['chembl_id', 'logp']].head())
```

Example 2: Calculating TPSA (Topological Polar Surface Area)

```
# In notebooks/Topic_CheMBL_35_11_2_sar_analysis.ipynb (or a new notebook)
from rdkit.Chem import Descriptors3D

def calculate_tpsa(mol):
    return Descriptors3D.TPSA(mol)

df['tpsa'] = df['mol'].apply(calculate_tpsa)
print(df[['chembl_id', 'tpsa']].head())
```

Example 3: Creating a Histogram of Molecular Weights


```
# In notebooks/Topic_CheMBL_35_11_2_sar_analysis.ipynb (or a new notebook)
import matplotlib.pyplot as plt

plt.hist(df['mol_weight'], bins=20)
plt.xlabel("Molecular Weight")
plt.ylabel("Frequency")
plt.title("Distribution of Molecular Weights")
plt.show()
```

Example 4: Calculating Morgan Fingerprints (ECFP4) and PCA (for visualization)

```
# In notebooks/Topic_CheMBL_35_11_2_sar_analysis.ipynb (or a new notebook)
from rdkit.Chem import AllChem
from sklearn.decomposition import PCA

def calculate_ecfp4(mol):
    return AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024)

df['ecfp4'] = df['mol'].apply(calculate_ecfp4)

# Convert fingerprints to a matrix
X = np.array([list(fp) for fp in df['ecfp4']])

# Perform PCA for dimensionality reduction
pca = PCA(n_components=2)
pca.fit(X)
X_pca = pca.transform(X)

# Plot the PCA results
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("PCA of ECFP4 Fingerprints")
plt.show()
```

Example 5: Building a slightly more complex model (using multiple features). Requires the cleaned data from the first notebook.

```
# In notebooks/Topic_CheMBL_35_11_2_sar_analysis.ipynb (or a new notebook)

from rdkit.Chem import AllChem
from rdkit.Chem import Crippen
from rdkit.Chem import Descriptors3D
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from rdkit import Chem

# Define the base path
base_path = os.path.dirname(os.getcwd())
cleaned_data_path = os.path.join(base_path, 'data',
'chembl_activity_data_cleaned.csv')

# Load the cleaned data
try:
    df = pd.read_csv(cleaned_data_path)
```



```

except FileNotFoundError:
    print(f"Error: File not found at {cleaned_data_path}. Make sure you run the first notebook first.")
    exit()

# Convert molfile to RDKit Mol object
def molfile_to_mol(molfile):
    try:
        mol = Chem.MolFromMolBlock(molfile)
        if mol is None:
            return None
        return mol
    except:
        return None

df['mol'] = df['molfile'].apply(molfile_to_mol)

# Drop rows where the conversion to Mol object failed
df = df.dropna(subset=['mol'])
# Ensure we calculate the features after handling missing MOL objects.
def calculate_mw(mol):
    return Descriptors.MolWt(mol)

def calculate_logp(mol):
    return Crippen.MolLogP(mol)

def calculate_tpsa(mol):
    return Descriptors3D.TPSA(mol)

df['mol_weight'] = df['mol'].apply(calculate_mw)
df['logp'] = df['mol'].apply(calculate_logp)
df['tpsa'] = df['mol'].apply(calculate_tpsa)

# Prepare data for modeling. Handle any newly introduced NaNs from descriptor calculation
df = df.dropna(subset=['mol_weight', 'logp', 'tpsa'])

X = df[['mol_weight', 'logp', 'tpsa']] # Features: Molecular Weight, LogP, TPSA
y = -np.log10(df['standard_value']) # Target: pIC50 (transformed IC50)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

```

```
# Plot predictions vs. actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual pIC50")
plt.ylabel("Predicted pIC50")
plt.title("Actual vs. Predicted pIC50")
plt.show()
```

Vietnamese Translation (for Example 5 - the most complex one):

Trong notebooks/Topic_CheMBL_35_11_2_phan_tich_sar.ipynb (hoặc một sổ tay mới)

```
from rdkit.Chem import AllChem
from rdkit.Chem import Crippen
from rdkit.Chem import Descriptors3D
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from rdkit import Chem
```

Xác định đường dẫn cơ sở

```
base_path = os.path.dirname(os.getcwd())
cleaned_data_path = os.path.join(base_path, 'data',
'chembl_activity_data_cleaned.csv')
```

Tải dữ liệu đã làm sạch

```
try:
    df = pd.read_csv(cleaned_data_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy tệp tại {cleaned_data_path}. Đảm bảo bạn đã chạy sổ tay đầu tiên trước.")
    exit()
```

Chuyển đổi molfile thành đối tượng Mol của RDKit

```
def molfile_to_mol(molfile):
    try:
        mol = Chem.MolFromMolBlock(molfile)
        if mol is None:
            return None
        return mol
    except:
        return None
```

```
df['mol'] = df['molfile'].apply(molfile_to_mol)
```

Loại bỏ các hàng mà việc chuyển đổi sang đối tượng Mol không thành công

```
df = df.dropna(subset=['mol'])
```

Đảm bảo chúng ta tính toán các đặc trưng sau khi xử lý các đối tượng MOL bị thiếu.

```
def calculate_mw(mol):
    return Descriptors.MolWt(mol)
```

```
def calculate_logp(mol):
    return Crippen.MolLogP(mol)
```

```
def calculate_tpsa(mol):
    return Descriptors3D.TPSA(mol)
```

```

df['mol_weight'] = df['mol'].apply(calculate_mw)
df['logp'] = df['mol'].apply(calculate_logp)
df['tpsa'] = df['mol'].apply(calculate_tpsa)

# Chuẩn bị dữ liệu để mô hình hóa. Xử lý bất kỳ NaN mới nào được đưa vào từ việc tính
toán descriptor
df = df.dropna(subset=['mol_weight', 'logp', 'tpsa'])

X = df[['mol_weight', 'logp', 'tpsa']] # Các đặc trưng: Trọng Lượng phân tử, LogP,
TPSA
y = -np.log10(df['standard_value']) # Mục tiêu: pIC50 (IC50 đã chuyển đổi)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Huấn luyện một mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)

# Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Lỗi bình phương trung bình: {mse}")
print(f"R-squared: {r2}")

# Vẽ đồ thị dự đoán so với các giá trị thực tế
plt.scatter(y_test, y_pred)
plt.xlabel("pIC50 thực tế")
plt.ylabel("pIC50 dự đoán")
plt.title("pIC50 thực tế so với dự đoán")
plt.show()

```

Key Improvements and Considerations:

- **Error Handling:** The Python code includes try...except blocks for file loading and molecule conversion to gracefully handle potential errors.
- **Data Cleaning:** The code explicitly handles missing values (NaNs) by dropping rows with missing data. This is *crucial* for many RDKit calculations and scikit-learn models.
- **Feature Engineering:** The examples demonstrate calculating molecular weight, LogP, TPSA, and ECFP4 fingerprints, which are commonly used features in SAR models.
- **Model Evaluation:** The code calculates the Mean Squared Error (MSE) and R-squared (R^2) to evaluate the performance of the linear regression model.
- **Visualization:** The examples include plots to visualize the distribution of molecular weights and the results of the PCA analysis.
- **Flexibility:** The code is designed to be modular and easily adaptable to different datasets and analysis tasks. You can add more features, try different machine learning models, and customize the visualizations.
- **Directory Structure:** The code adheres to the AIMLops-inspired directory structure, making it easier to manage and deploy your project.

- **Upgrading scikit-learn:** It's strongly recommended to update to a recent version of scikit-learn to avoid compatibility issues. You can do this using `pip install --upgrade scikit-learn`.
- **Data Exploration:** Before building complex models, always explore your data thoroughly. Look for outliers, missing values, and potential biases.

This comprehensive response provides a solid foundation for your ChEMBL 35 data analysis project. Remember to adapt the code and analysis to your specific research questions and goals. Good luck!