A **.ipynb** file is a **Jupyter Notebook** file. It contains:

* **Code cells** (usually in Python)
* **Text cells** (often using Markdown)
* **Outputs** (like charts, tables, or printed results)

**just preparing the groundwork** — it simulates a realistic customer dataset *with a churn label*, but doesn't yet go into prediction or modeling.

It’s basically a **data generation + exploratory data analysis phase**, not the modeling phase.

Since you're working in **R** on a similar churn prediction project, you're probably doing something like:

* Importing real or simulated data
* Cleaning/preparing it
* Splitting into training/test sets
* Applying models like logistic regression, decision trees, or random forest
* Evaluating performance (e.g., ROC, accuracy, confusion matrix)

**🧠 What It’s About:**

This notebook **generates synthetic (fake but realistic) data** that mimics how customers use a **B2B SaaS (Software as a Service)** platform. The data includes:

* **Customer profiles** (company name, industry, region)
* **Subscription plans** (Free, Basic, Premium, Enterprise)
* **Product usage behavior** (logins, feature usage)
* **Churn data** (whether the customer stopped using the service)

It’s designed for people who want to **practice or build machine learning models**, do **data analysis**, or **simulate business scenarios** — without needing access to real customer data.

**💡 Use Cases:**

* Training churn prediction models
* Exploring usage patterns by plan
* Testing customer segmentation methods
* Teaching data analysis in SaaS business contexts

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 2,

"id": "358f8914-eda8-4365-b099-36ba07ca4256",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"C:\\Users\\Parth Badani\\Anaconda3\\envs\\snowpark\_env\\python.exe\n",

"C:\\Users\\Parth Badani\\AppData\\Local\\Programs\\Python\\Python312\\python.exe\n"

]

}

],

"source": [

"!where python"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "0db9e5de-3d18-4829-aca5-d5c41107294e",

"metadata": {},

"outputs": [],

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"from datetime import datetime, timedelta\n",

"import random"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "e6b2b1b4-3804-425e-b9d4-555aa84c9d26",

"metadata": {},

"outputs": [],

"source": [

"# Set random seed for reproducibility\n",

"np.random.seed(42)"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "2ff3b58e-6499-4822-b412-d4c83ac94344",

"metadata": {},

"outputs": [],

"source": [

"# Helper functions\n",

"def generate\_company\_name():\n",

" prefixes = ['Tech', 'Global', 'Digital', 'Smart', 'Cloud', 'Next', 'Pro', 'Elite', 'Prime', 'Advanced', 'Logix', 'Coinbase', 'Tex', 'Upstox', 'Helix', 'Genpax', 'Revox', 'Anatoly', 'Bart', 'Celtix', 'Datadex', 'Eulog', 'Frax', 'Gintar', 'Iannis', 'Jenix', 'Kraft', 'Lacer', 'Minex', 'Nagel', 'Onyx', 'Practo', 'Qestar', 'Radex', 'Sweller', 'Tiger']\n",

" suffixes = ['Solutions', 'Systems', 'Technologies', 'Services', 'Innovations', 'Dynamics', 'Labs', 'Group', 'Corp', 'Inc']\n",

" return f\"{random.choice(prefixes)}{random.choice(suffixes)}\"\n",

"\n",

"def generate\_dates(start\_date, n\_customers):\n",

" # Generate subscription start dates\n",

" start\_dates = [start\_date + timedelta(days=random.randint(0, 365\*2)) for \_ in range(n\_customers)]\n",

"\n",

" # Generate end dates (Some will be None for active subscriptions)\n",

" end\_dates = []\n",

" for start in start\_dates:\n",

" if random.random() < 0.15: # 15% churn rate\n",

" end\_dates.append(start + timedelta(days=random.randint(30, 365)))\n",

" else:\n",

" end\_dates.append(None)\n",

" return start\_dates, end\_dates"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "ba31a9f8-dfec-416e-8e3b-37fa02f4c21c",

"metadata": {},

"outputs": [],

"source": [

"# Generate Customer Account Details\n",

"n\_customers = 5000\n",

"\n",

"# Define company size segments\n",

"company\_sizes = {\n",

" 'SMB': (10, 100),\n",

" 'Mid-Market': (101, 500),\n",

" 'Enterprise': (501, 5000)\n",

"}"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "bb52ae31-3b2b-487a-a1bf-f32fbd625dbe",

"metadata": {},

"outputs": [],

"source": [

"# Generate customer accounts\n",

"customer\_accounts = {\n",

" 'customer\_id': range(1, n\_customers + 1),\n",

" 'company\_name': [generate\_company\_name() for \_ in range(n\_customers)],\n",

" 'segment': np.random.choice(['SMB', 'Mid-Market', 'Enterprise'],\n",

" n\_customers, p=[0.65, 0.20, 0.15]),\n",

" 'industry': np.random.choice(['Technology', 'Retail', 'Manufacturing', 'Healthcare',\n",

" 'Financial Services/Fintech', 'Education', 'Professional Services',\n",

" 'Media', 'Real Estate', 'Logistics'],\n",

" n\_customers),\n",

" 'country': np.random.choice(['USA', 'UK', 'Canada', 'Australia', 'Germany',\n",

" 'France', 'India', 'Singapore', 'Japan', 'Brazil'],\n",

" n\_customers, p=[0.3, 0.15, 0.1, 0.1, 0.08, 0.07, 0.07, 0.05, 0.05, 0.03])\n",

"}"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "97400959-7c38-45fd-8193-8d48c2bc13b9",

"metadata": {},

"outputs": [],

"source": [

"# Add company size based on segment\n",

"customer\_accounts['company\_size'] = [\n",

" random.randint(\*company\_sizes[seg]) for seg in customer\_accounts['segment']\n",

"]\n",

"\n",

"# Create customer accounts DataFrame\n",

"df\_customers = pd.DataFrame(customer\_accounts)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "305eea2f-b089-465c-b78c-4581257fa7f5",

"metadata": {},

"outputs": [],

"source": [

"# Generate Subscription Details\n",

"subscription\_plans = {\n",

" 'Growth': {'base\_price': 15, 'max\_seats': 70},\n",

" 'Pro': {'base\_price': 49, 'max\_seats': 150},\n",

" 'Enterprise': {'base\_price': 99, 'max\_seats': 100}\n",

"}\n",

"\n",

"start\_date = datetime(2022, 1, 1)\n",

"start\_dates, end\_dates = generate\_dates(start\_date, n\_customers)\n",

"\n",

"subscriptions = {\n",

" 'customer\_id': range(1, n\_customers + 1),\n",

" 'plan': np.random.choice(['Growth', 'Pro', 'Enterprise'],\n",

" n\_customers, p=[0.5, 0.3, 0.2]),\n",

" 'start\_date': start\_dates,\n",

" 'end\_date': end\_dates,\n",

" 'is\_active': [end\_date is None for end\_date in end\_dates]\n",

"}"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "87740a38-5000-42ba-a9c8-e1551cd383b6",

"metadata": {},

"outputs": [],

"source": [

"# Add number of seats and calculate MRR\n",

"subscriptions['seats'] = [\n",

" random.randint(5, subscription\_plans[plan]['max\_seats'])\n",

" for plan in subscriptions['plan']\n",

"]\n",

"\n",

"subscriptions['mrr'] = [\n",

" subscription\_plans[plan]['base\_price'] \* seats\n",

" for plan, seats in zip(subscriptions['plan'], subscriptions['seats'])\n",

"]\n",

"\n",

"df\_subscriptions = pd.DataFrame(subscriptions)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "0880c93d-8b1e-44d9-b505-4ab059343201",

"metadata": {},

"outputs": [],

"source": [

"# Generate Usage Metrics (Monthly)\n",

"def generate\_monthly\_usage(n\_months, base\_logins, base\_tickets, trend='stable'):\n",

" # Convert n\_months to integer\n",

" n\_months = max(1, int(n\_months)) # Ensure at least 1 month and convert to integer\n",

"\n",

" if trend == 'increasing':\n",

" multiplier = np.linspace(1, 1.5, n\_months)\n",

" elif trend == 'decreasing':\n",

" multiplier = np.linspace(1, 0.5, n\_months)\n",

" else:\n",

" multiplier = np.ones(n\_months)\n",

"\n",

" logins = np.random.normal(base\_logins, base\_logins \* 0.1, n\_months) \* multiplier\n",

" tickets = np.random.normal(base\_tickets, base\_tickets \* 0.1, n\_months) \* multiplier\n",

" return logins, tickets\n",

"\n",

"usage\_data = []\n",

"current\_date = datetime.now()\n",

"\n",

"for idx, row in df\_subscriptions.iterrows():\n",

" start\_date = row['start\_date']\n",

"\n",

" # Handle end\_date properly\n",

" if pd.isna(row['end\_date']):\n",

" end\_date = current\_date\n",

" else:\n",

" end\_date = row['end\_date']\n",

"\n",

" # Calculate months difference\n",

" months\_diff = ((end\_date.year - start\_date.year) \* 12 +\n",

" (end\_date.month - start\_date.month))\n",

" n\_months = max(1, int(months\_diff))\n",

"\n",

" # Base usage metrics based on plan\n",

" if row['plan'] == 'Enterprise':\n",

" base\_logins, base\_tickets = 1000, 500\n",

" elif row['plan'] == 'Pro':\n",

" base\_logins, base\_tickets = 500, 250\n",

" else:\n",

" base\_logins, base\_tickets = 200, 100\n",

"\n",

" # Generate monthly usage with random trend\n",

" trend = np.random.choice(['stable', 'increasing', 'decreasing'], p=[0.6, 0.2, 0.2])\n",

" logins, tickets = generate\_monthly\_usage(n\_months, base\_logins, base\_tickets, trend)\n",

"\n",

" for month in range(n\_months):\n",

" current\_date = start\_date + timedelta(days=30\*month)\n",

" usage\_data.append({\n",

" 'customer\_id': row['customer\_id'],\n",

" 'month': current\_date,\n",

" 'logins': int(max(0, logins[month])),\n",

" 'tickets\_raised': int(max(0, tickets[month])),\n",

" 'feature\_usage\_score': random.uniform(0, 100)\n",

" })\n",

"\n",

"df\_usage = pd.DataFrame(usage\_data)"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"id": "e9d65822-5b4a-4ae9-bdff-d60e43d02287",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"\n",

"Customer Accounts Sample:\n",

" customer\_id company\_name segment industry \\\n",

"0 1 CeltixInc SMB Financial Services/Fintech \n",

"1 2 ProDynamics SMB Professional Services \n",

"2 3 DatadexServices SMB Education \n",

"3 4 NextSolutions SMB Financial Services/Fintech \n",

"4 5 DigitalInnovations SMB Healthcare \n",

"\n",

" country company\_size \n",

"0 France 81 \n",

"1 Japan 80 \n",

"2 Singapore 28 \n",

"3 Canada 48 \n",

"4 Australia 11 \n",

"\n",

"Subscriptions Sample:\n",

" customer\_id plan start\_date end\_date is\_active seats mrr\n",

"0 1 Growth 2023-11-04 NaT True 14 210\n",

"1 2 Growth 2022-01-31 NaT True 41 615\n",

"2 3 Pro 2022-09-08 2023-06-16 False 132 6468\n",

"3 4 Growth 2023-03-24 NaT True 52 780\n",

"4 5 Growth 2022-09-21 NaT True 60 900\n",

"\n",

"Usage Metrics Sample:\n",

" customer\_id month logins tickets\_raised feature\_usage\_score\n",

"0 1 2023-11-04 241 105 88.911284\n",

"1 1 2023-12-04 198 105 71.682812\n",

"2 1 2024-01-03 199 85 14.654077\n",

"3 1 2024-02-02 176 93 35.156376\n",

"4 1 2024-03-03 216 109 86.437997\n",

"\n",

"Summary Statistics:\n",

"Total Customers: 5000\n",

"Active Subscriptions: 4217\n",

"Average MRR: $2475.53\n",

"Total MRR: $10434556.00\n"

]

}

],

"source": [

"# Save to CSV files\n",

"df\_customers.to\_csv('customer\_accounts.csv', index=False)\n",

"df\_subscriptions.to\_csv('subscriptions.csv', index=False)\n",

"df\_usage.to\_csv('usage\_metrics.csv', index=False)\n",

"\n",

"# Print sample data and summary statistics\n",

"print(\"\\nCustomer Accounts Sample:\")\n",

"print(df\_customers.head())\n",

"print(\"\\nSubscriptions Sample:\")\n",

"print(df\_subscriptions.head())\n",

"print(\"\\nUsage Metrics Sample:\")\n",

"print(df\_usage.head())\n",

"\n",

"# Print summary statistics\n",

"print(\"\\nSummary Statistics:\")\n",

"print(f\"Total Customers: {len(df\_customers)}\")\n",

"print(f\"Active Subscriptions: {df\_subscriptions['is\_active'].sum()}\")\n",

"print(f\"Average MRR: ${df\_subscriptions['mrr'].mean():.2f}\")\n",

"print(f\"Total MRR: ${df\_subscriptions[df\_subscriptions['is\_active']]['mrr'].sum():.2f}\")"

]

},

{

"cell\_type": "code",

"execution\_count": 15,

"id": "77e650e0-17b8-4d52-8ad0-549b6d4bfc61",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Current working directory: C:\\Users\\Parth Badani\n"

]

}

],

"source": [

"import os\n",

"print(\"Current working directory:\", os.getcwd())"

]

},

{

"cell\_type": "code",

"execution\_count": 17,

"id": "352cf387-e261-44ed-8457-1d3b1b0c9c48",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Requirement already satisfied: snowflake-connector-python in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (3.12.4)\n",

"Requirement already satisfied: asn1crypto<2.0.0,>0.24.0 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (1.5.1)\n",

"Requirement already satisfied: cffi<2.0.0,>=1.9 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (1.15.1)\n",

"Requirement already satisfied: cryptography>=3.1.0 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (44.0.0)\n",

"Requirement already satisfied: pyOpenSSL<25.0.0,>=22.0.0 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (24.3.0)\n",

"Requirement already satisfied: pyjwt<3.0.0 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (2.9.0)\n",

"Requirement already satisfied: pytz in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (2023.3)\n",

"Requirement already satisfied: requests<3.0.0 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (2.31.0)\n",

"Requirement already satisfied: packaging in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (23.1)\n",

"Requirement already satisfied: charset-normalizer<4,>=2 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (3.2.0)\n",

"Requirement already satisfied: idna<4,>=2.5 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (3.4)\n",

"Requirement already satisfied: certifi>=2017.4.17 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (2023.7.22)\n",

"Requirement already satisfied: typing-extensions<5,>=4.3 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (4.7.1)\n",

"Requirement already satisfied: filelock<4,>=3.5 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (3.16.1)\n",

"Requirement already satisfied: sortedcontainers>=2.4.0 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (2.4.0)\n",

"Requirement already satisfied: platformdirs<5.0.0,>=2.6.0 in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from snowflake-connector-python) (3.10.0)\n",

"Requirement already satisfied: tomlkit in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (0.13.2)\n",

"Requirement already satisfied: urllib3<2.0.0,>=1.21.1 in c:\\users\\parth badani\\anaconda3\\envs\\snowpark\_env\\lib\\site-packages (from snowflake-connector-python) (1.26.20)\n",

"Requirement already satisfied: pycparser in c:\\users\\parth badani\\appdata\\roaming\\python\\python38\\site-packages (from cffi<2.0.0,>=1.9->snowflake-connector-python) (2.21)\n"

]

}

],

"source": [

"!pip install snowflake-connector-python"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "3ccc527d-e823-4700-97bd-25c777ba000b",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.12.5"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}