Customer Churn Model Using XGBoost Framework

1. Customer Retention Retail Dataset

This dataset can be used to understand what are the various marketing strategy based on consumer behaviour that can be adopted to increase customer retention of a retail store.

An online tea retail store which sells tea of different flavors across various cities in India. The dataset contains data about the store's customers, their orders, quantity ordered, order frequency, city,etc. This is a large dataset which will help in analysis.

Reference: https://www.kaggle.com/uttamp/store-data

In [ ]:
```
%%html
<style>
table {float:left}
</style>
```

| column | Description |--|--| | custid | Computer generated ID to identify customers throughout the database |retained | 1, if customer is assumed to be active, 0 = otherwise |created | Date when the contact was created in the database - when the customer joined |firstorder | Date when the customer placed first order |lastorder | Date when the customer placed last order |esent | Number of emails sent |eopenrate | Number of emails opened divided by number of emails sent |eclickrate | Number of emails clicked divided by number of emails sent |avgorder | Average order size for the customer |ordfreq | Number of orders divided by customer tenure |paperless | 1 if customer subscribed for paperless communication (only online) |refill | 1 if customer subscribed for automatic refill |doorstep | 1 if customer subscribed for doorstep delivery |train | 1 if customer is in the training database |favday | Customer's favorite delivery day |city | City where the customer resides in

2. Import Packages and Constants

Install shap and smdebug packages if not already installed and restart kernel after installing the packages

In [ ]:
```
import re
import s3fs
import shap
import time
import boto3
import pandas as pd
import numpy as np

from itertools import islice
import matplotlib.pyplot as plt
```

```
import sagemaker
from sagemaker.xgboost.estimator import XGBoost
from sagemaker.session import Session
from sagemaker.inputs import TrainingInput
from sagemaker.debugger import DebuggerHookConfig,CollectionConfig
from sagemaker.debugger import rule_configs, Rule
from smdebug.trials import create_trial
from sagemaker.tuner import (
    IntegerParameter,
    CategoricalParameter,
    ContinuousParameter,
    HyperparameterTuner
)
```

In [ ]:
```
#Replace this value with the S3 Bucket Created

default_bucket = "sagemaker-studio-314146328250-gtb7x2llho5"
```

In [ ]:
```
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
region = sagemaker_session.boto_region_name
```

3. Preprocess Data

In [ ]:
```
def preprocess_data(file_path):
    df = pd.read_csv(file_path)
    ## Convert to datetime columns
    df["firstorder"]=pd.to_datetime(df["firstorder"],errors='coerce')
    df["lastorder"] = pd.to_datetime(df["lastorder"],errors='coerce')
    ## Drop Rows with null values
    df = df.dropna()
    ## Create Column which gives the days between the last order and the first orde
    df["first_last_days_diff"] = (df['lastorder']-df['firstorder']).dt.days
    ## Create Column which gives the days between when the customer record was crea
    df['created'] = pd.to_datetime(df['created'])
    df['created_first_days_diff']=(df['created']-df['firstorder']).dt.days
    ## Drop Columns
    df.drop(['custid','created','firstorder','lastorder'],axis=1,inplace=True)
    ## Apply one hot encoding on favday and city columns
    df = pd.get_dummies(df,prefix=['favday','city'],columns=['favday','city'])
    return df
```

In [ ]:
```
storedata = preprocess_data(f"s3://{default_bucket}/data/storedata_total.csv")
```

In [ ]:
```
storedata.head()
```

Out[ ]:

	retained	esent	eopenrate	eclickrate	avgorder	ordfreq	paperless	refill	doorstep
0	0	29	100.000000	3.448276	14.52	0.000000	0	0	0
1	1	95	92.631579	10.526316	83.69	0.181641	1	1	1
2	0	0	0.000000	0.000000	33.58	0.059908	0	0	0
3	0	0	0.000000	0.000000	54.96	0.000000	0	0	0
4	1	30	90.000000	13.333333	111.91	0.008850	0	0	0

5 rows × 22 columns

4. Split Train, Test and Validation Datasets

In [ ]:
```
def split_datasets(df):
    y=df.pop("retained")
    X_pre = df
    y_pre = y.to_numpy().reshape(len(y),1)
    feature_names = list(X_pre.columns)
    X= np.concatenate((y_pre,X_pre),axis=1)
    np.random.shuffle(X)
    train,validation,test=np.split(X,[int(.7*len(X)),int(.85*len(X))])
    return feature_names,train,validation,test
```

In [ ]:
```
feature_names,train,validation,test = split_datasets(storedata)
```

In [ ]:
```
pd.DataFrame(train).to_csv(f"s3://{default_bucket}/data/train/train.csv",header=Fal
pd.DataFrame(validation).to_csv(f"s3://{default_bucket}/data/validation/validation.
pd.DataFrame(test).to_csv(f"s3://{default_bucket}/data/test/test.csv",header=False,
```

5. Hyperparameter Tuning HPO

In [ ]:
```
s3_input_train = TrainingInput(
    s3_data=f"s3://{default_bucket}/data/train/",content_type="csv")
s3_input_validation = TrainingInput(
    s3_data=f"s3://{default_bucket}/data/validation/",content_type="csv")
```

In [ ]:
```
fixed_hyperparameters = {
    "eval_metric":"auc",
    "objective":"binary:logistic",
    "num_round":"100",
    "rate_drop":"0.3",
    "tweedie_variance_power":"1.4"
}
```

In [ ]:
```
sess = sagemaker.Session()
container = sagemaker.image_uris.retrieve("xgboost",region,"0.90-2")

estimator = sagemaker.estimator.Estimator(
```

```
    container,
    role,
    instance_count=1,
    hyperparameters=fixed_hyperparameters,
    instance_type="ml.m4.xlarge",
    output_path="s3://{}/output".format(default_bucket),
    sagemaker_session=sagemaker_session
)
```

```
[01/08/25 04:02:34] INFO     Defaulting to only available Python version: py3
                    INFO     Defaulting to only supported image scope: cpu.
```

In [ ]:
```
hyperparameter_ranges = {
    "eta": ContinuousParameter(0, 1),
    "min_child_weight": ContinuousParameter(1, 10),
    "alpha": ContinuousParameter(0, 2),
    "max_depth": IntegerParameter(1, 10),
}
```

In [ ]:
```
objective_metric_name = "validation:auc"
```

In [ ]:
```
tuner = HyperparameterTuner(
    estimator,objective_metric_name,hyperparameter_ranges,max_jobs=10,max_parallel_
```

In [ ]:
```
tuner.fit({
    "train":s3_input_train,
    "validation":s3_input_validation
    },include_cls_metadata=False)
```

```
                    WARNING  No finished training job found associated with t
                             Please make sure this estimator is only used for
                             config
                    WARNING  No finished training job found associated with t
                             Please make sure this estimator is only used for
                             config
                    INFO     Creating hyperparameter tuning job with name:
                             sagemaker-xgboost-250108-0402
.....................................................................!
```

In [ ]:
```
tuning_job_result = boto3.client("sagemaker").describe_hyper_parameter_tuning_job(
    HyperParameterTuningJobName=tuner.latest_tuning_job.job_name
)
```

In [ ]:
```
job_count = tuning_job_result["TrainingJobStatusCounters"]["Completed"]
print("%d training jobs have completed" %job_count)
```

```
10 training jobs have completed
```

In [ ]:
```
from pprint import pprint

if tuning_job_result.get("BestTrainingJob",None):
    print("Best Model found so far:")
    pprint(tuning_job_result["BestTrainingJob"])
else:
    print("No training jobs have reported results yet.")
```

```
Best Model found so far:
{'CreationTime': datetime.datetime(2025, 1, 8, 4, 7, 39, tzinfo=tzlocal()),
 'FinalHyperParameterTuningJobObjectiveMetric': {'MetricName': 'validation:auc',
                                                 'Value': 0.97816002368927},
 'ObjectiveStatus': 'Succeeded',
 'TrainingEndTime': datetime.datetime(2025, 1, 8, 4, 8, 13, tzinfo=tzlocal()),
 'TrainingJobArn': 'arn:aws:sagemaker:us-east-1:314146328250:training-job/sagemaker-
xgboost-250108-0402-009-dd996545',
 'TrainingJobName': 'sagemaker-xgboost-250108-0402-009-dd996545',
 'TrainingJobStatus': 'Completed',
 'TrainingStartTime': datetime.datetime(2025, 1, 8, 4, 7, 44, tzinfo=tzlocal()),
 'TunedHyperParameters': {'alpha': '1.9316797103099645',
                          'eta': '0.09827721370039',
                          'max_depth': '4',
                          'min_child_weight': '8.498603164205788'}}
```

In [ ]:
```
best_hyperparameters = tuning_job_result["BestTrainingJob"]["TunedHyperParameters"]
```

In [ ]:
```
best_hyperparameters
```

Out[ ]:
```
{'alpha': '1.9316797103099645',
 'eta': '0.09827721370039',
 'max_depth': '4',
 'min_child_weight': '8.498603164205788'}
```

7. XGBoost Model with SageMaker Debugger

In [ ]:
```
hyperparameters = {**fixed_hyperparameters,**best_hyperparameters}
save_interval = 5
base_job_name = "demo-smdebug-xgboost-churn-classification"
```

In [ ]:
```
container = sagemaker.image_uris.retrieve("xgboost",region,"0.90-2")
```

```
[01/08/25 04:08:39] INFO     Defaulting to only available Python version: py3
                    INFO     Defaulting to only supported image scope: cpu.
```

In [ ]:
```
estimator = sagemaker.estimator.Estimator(
    container,
    role,
    base_job_name=base_job_name,
    instance_count=1,
    instance_type="ml.m4.xlarge",
    output_path="s3://{}/output".format(default_bucket),
    sagemaker_session=sess,
```

```
    hyperparameters=hyperparameters,
    max_run=1800,
    debugger_hook_config = DebuggerHookConfig(
        s3_output_path=f"s3://{default_bucket}/debugger/",  # Required
        collection_configs=[
            CollectionConfig(
                name="metrics",
                parameters={
                    "save_interval": "5"
                }),
            CollectionConfig(
                name="feature_importance", parameters={"save_interval": "5"}
            ),
            CollectionConfig(name="full_shap", parameters={"save_interval": "5"}),
            CollectionConfig(name="average_shap", parameters={"save_interval": "5"}
        ]
    ),
    rules=[
        Rule.sagemaker(
            rule_configs.loss_not_decreasing(),
            rule_parameters={
                "collection_names": "metrics",
                "num_steps": "10",
            },
        ),
    ]
)
```

In [ ]:
```
estimator.fit(
        {"train":s3_input_train,"validation":s3_input_validation},wait=False
    )
```

```
INFO     SageMaker Python SDK will collect telemetry to h
         understand our user's needs, diagnose issues, an
         additional features.
         To opt out of telemetry, please disable via Tele
         parameter in SDK defaults config. For more infor
         to
         https://sagemaker.readthedocs.io/en/stable/overv
         guring-and-using-defaults-with-the-sagemaker-pyt

INFO     Ignoring unnecessary instance type: None.

INFO     Creating training-job with name:
         demo-smdebug-xgboost-churn-classificati-2025-01-
```

In [ ]:
```
for _ in range(36):
    job_name = estimator.latest_training_job.name
    client = estimator.sagemaker_session.sagemaker_client
    description = client.describe_training_job(TrainingJobName=job_name)
    training_job_status = description["TrainingJobStatus"]
    rule_job_summary = estimator.latest_training_job.rule_job_summary()
    rule_evaluation_status = rule_job_summary[0]["RuleEvaluationStatus"]
    print(
```

```
        "Training job status: {}, Rule Evaluation Status: {}".format(
            training_job_status, rule_evaluation_status
        )
    )
    if training_job_status in ["Completed", "Failed"]:
        break
    time.sleep(10)
```

```
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: InProgress, Rule Evaluation Status: InProgress
Training job status: Completed, Rule Evaluation Status: InProgress
```

8. Analyze Debugger Output

In [ ]:
```
estimator.latest_training_job.rule_job_summary()
```

Out[ ]:
```
[{'RuleConfigurationName': 'LossNotDecreasing',
  'RuleEvaluationJobArn': 'arn:aws:sagemaker:us-east-1:314146328250:processing-job/demo-smdebug-xgboost-churn-LossNotDecreasing-1f526cab',
  'RuleEvaluationStatus': 'InProgress',
  'LastModifiedTime': datetime.datetime(2025, 1, 8, 4, 11, 23, 617000, tzinfo=tzlocal())}]
```

In [ ]:
```
s3_output_path = estimator.latest_job_debugger_artifacts_path()
trial = create_trial(s3_output_path)
```

```
[2025-01-08 04:11:42.100 default:496 INFO s3_trial.py:42] Loading trial debug-output at path s3://sagemaker-studio-314146328250-gtb7x2llho5/debugger/demo-smdebug-xgboost-churn-classificati-2025-01-08-04-08-39-695/debug-output
```

In [ ]:
```
trial.tensor_names()
```

```
[2025-01-08 04:11:43.045 default:496 INFO trial.py:197] Training has ended, will refresh one final time in 1 sec.
[2025-01-08 04:11:44.066 default:496 INFO trial.py:210] Loaded all steps
```

Out[ ]:
```
['average_shap/f0',
 'average_shap/f1',
 'average_shap/f10',
 'average_shap/f11',
 'average_shap/f12',
 'average_shap/f13',
 'average_shap/f14',
 'average_shap/f15',
 'average_shap/f16',
 'average_shap/f17',
 'average_shap/f18',
 'average_shap/f19',
 'average_shap/f2',
 'average_shap/f20',
 'average_shap/f3',
 'average_shap/f4',
 'average_shap/f5',
 'average_shap/f6',
 'average_shap/f7',
 'average_shap/f8',
 'average_shap/f9',
 'feature_importance/cover/f0',
 'feature_importance/cover/f1',
 'feature_importance/cover/f2',
 'feature_importance/cover/f3',
 'feature_importance/cover/f4',
 'feature_importance/cover/f5',
 'feature_importance/cover/f6',
 'feature_importance/cover/f7',
 'feature_importance/cover/f8',
 'feature_importance/cover/f9',
 'feature_importance/gain/f0',
 'feature_importance/gain/f1',
 'feature_importance/gain/f2',
 'feature_importance/gain/f3',
 'feature_importance/gain/f4',
 'feature_importance/gain/f5',
 'feature_importance/gain/f6',
 'feature_importance/gain/f7',
 'feature_importance/gain/f8',
 'feature_importance/gain/f9',
 'feature_importance/total_cover/f0',
 'feature_importance/total_cover/f1',
 'feature_importance/total_cover/f2',
 'feature_importance/total_cover/f3',
 'feature_importance/total_cover/f4',
 'feature_importance/total_cover/f5',
 'feature_importance/total_cover/f6',
 'feature_importance/total_cover/f7',
 'feature_importance/total_cover/f8',
 'feature_importance/total_cover/f9',
 'feature_importance/total_gain/f0',
 'feature_importance/total_gain/f1',
 'feature_importance/total_gain/f2',
 'feature_importance/total_gain/f3',
 'feature_importance/total_gain/f4',
```

```
 'feature_importance/total_gain/f5',
 'feature_importance/total_gain/f6',
 'feature_importance/total_gain/f7',
 'feature_importance/total_gain/f8',
 'feature_importance/total_gain/f9',
 'feature_importance/weight/f0',
 'feature_importance/weight/f1',
 'feature_importance/weight/f2',
 'feature_importance/weight/f3',
 'feature_importance/weight/f4',
 'feature_importance/weight/f5',
 'feature_importance/weight/f6',
 'feature_importance/weight/f7',
 'feature_importance/weight/f8',
 'feature_importance/weight/f9',
 'full_shap/f0',
 'full_shap/f1',
 'full_shap/f10',
 'full_shap/f11',
 'full_shap/f12',
 'full_shap/f13',
 'full_shap/f14',
 'full_shap/f15',
 'full_shap/f16',
 'full_shap/f17',
 'full_shap/f18',
 'full_shap/f19',
 'full_shap/f2',
 'full_shap/f20',
 'full_shap/f3',
 'full_shap/f4',
 'full_shap/f5',
 'full_shap/f6',
 'full_shap/f7',
 'full_shap/f8',
 'full_shap/f9',
 'train-auc',
 'validation-auc']
```

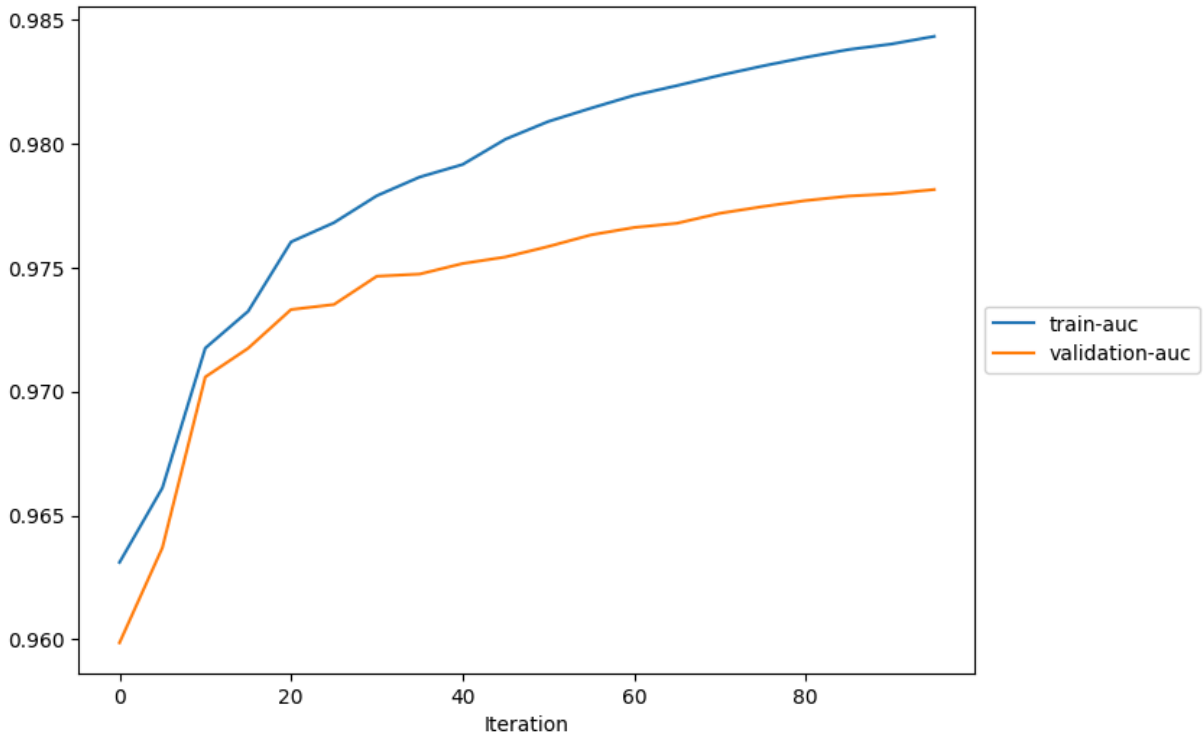
In [ ]:
```
trial.tensor("average_shap/f1").values()
```

Out[ ]:
```
{0: array([-7.441343e-05], dtype=float32),
 5: array([-0.00030935], dtype=float32),
 10: array([0.00310442], dtype=float32),
 15: array([0.00522607], dtype=float32),
 20: array([0.00574773], dtype=float32),
 25: array([0.01010881], dtype=float32),
 30: array([0.01123486], dtype=float32),
 35: array([0.01256887], dtype=float32),
 40: array([0.01193235], dtype=float32),
 45: array([0.02519895], dtype=float32),
 50: array([0.04025906], dtype=float32),
 55: array([0.05220644], dtype=float32),
 60: array([0.05429957], dtype=float32),
 65: array([0.06214123], dtype=float32),
 70: array([0.07114397], dtype=float32),
 75: array([0.07868035], dtype=float32),
 80: array([0.07873096], dtype=float32),
 85: array([0.07926091], dtype=float32),
 90: array([0.07749772], dtype=float32),
 95: array([0.08647537], dtype=float32)}
```

In [ ]:
```
MAX_PLOTS = 35


def get_data(trial, tname):
    """
    For the given tensor name, walks though all the iterations
    for which you have data and fetches the values.
    Returns the set of steps and the values.
    """
    tensor = trial.tensor(tname)
    steps = tensor.steps()
    vals = [tensor.value(s) for s in steps]
    return steps, vals


def match_tensor_name_with_feature_name(tensor_name, feature_names=feature_names):
    feature_tag = tensor_name.split("/")
    for ifeat, feature_name in enumerate(feature_names):
        if feature_tag[-1] == "f{}".format(str(ifeat)):
            return feature_name
    return tensor_name


def plot_collection(trial, collection_name, regex=".*", figsize=(8, 6)):
    """
    Takes a `trial` and a collection name, and
    plots all tensors that match the given regex.
    """
    fig, ax = plt.subplots(figsize=figsize)
    tensors = trial.collection(collection_name).tensor_names
    matched_tensors = [t for t in tensors if re.match(regex, t)]
    for tensor_name in islice(matched_tensors, MAX_PLOTS):
        steps, data = get_data(trial, tensor_name)
        ax.plot(steps, data, label=match_tensor_name_with_feature_name(tensor_name)
```

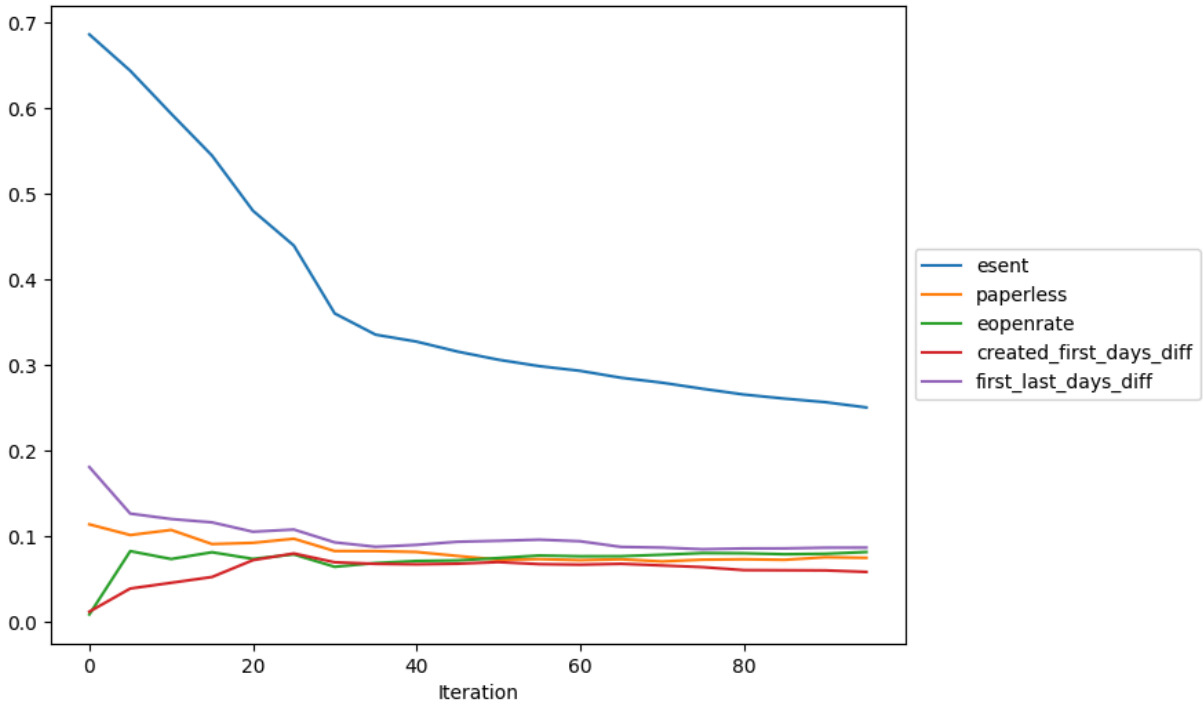
```
    ax.legend(loc="center left", bbox_to_anchor=(1, 0.5))
    ax.set_xlabel("Iteration")
```

In [ ]:
```
plot_collection(trial, "metrics")
```



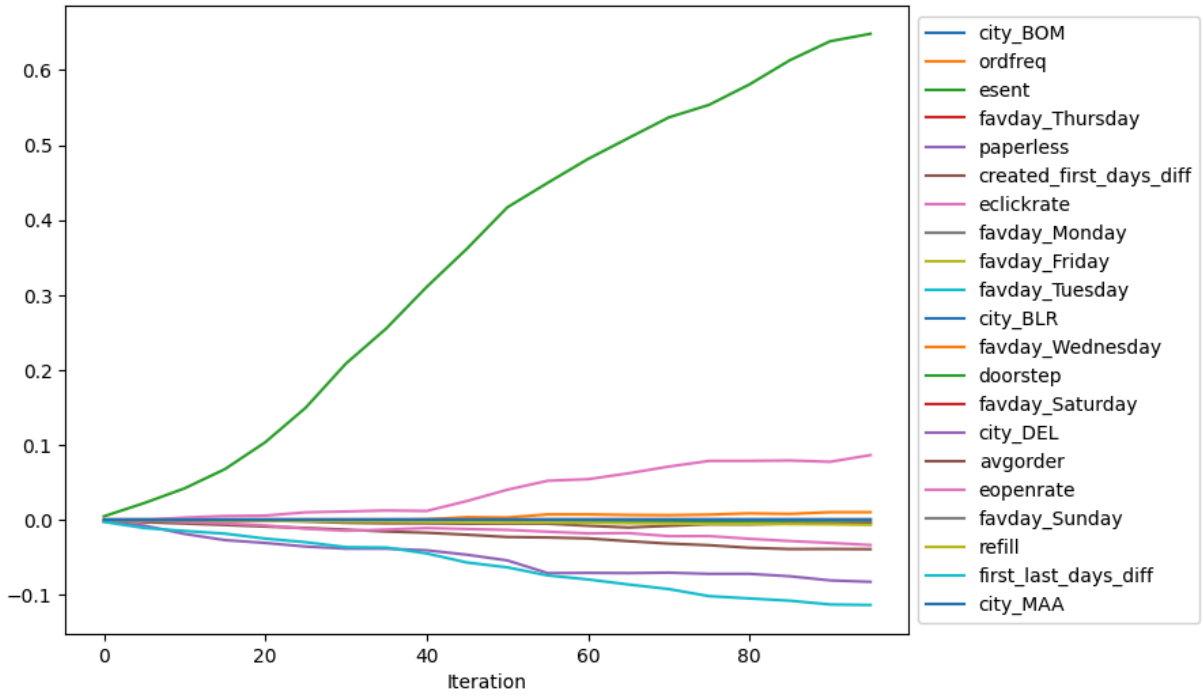
In [ ]:
```
def plot_feature_importance(trial, importance_type="weight"):
    SUPPORTED_IMPORTANCE_TYPES = ["weight", "gain", "cover", "total_gain", "total_c
    if importance_type not in SUPPORTED_IMPORTANCE_TYPES:
        raise ValueError(f"{importance_type} is not one of the supported importance
    plot_collection(trial, "feature_importance", regex=f"feature_importance/{import
```

In [ ]:
```
plot_feature_importance(trial, importance_type="cover")
```



SHAP

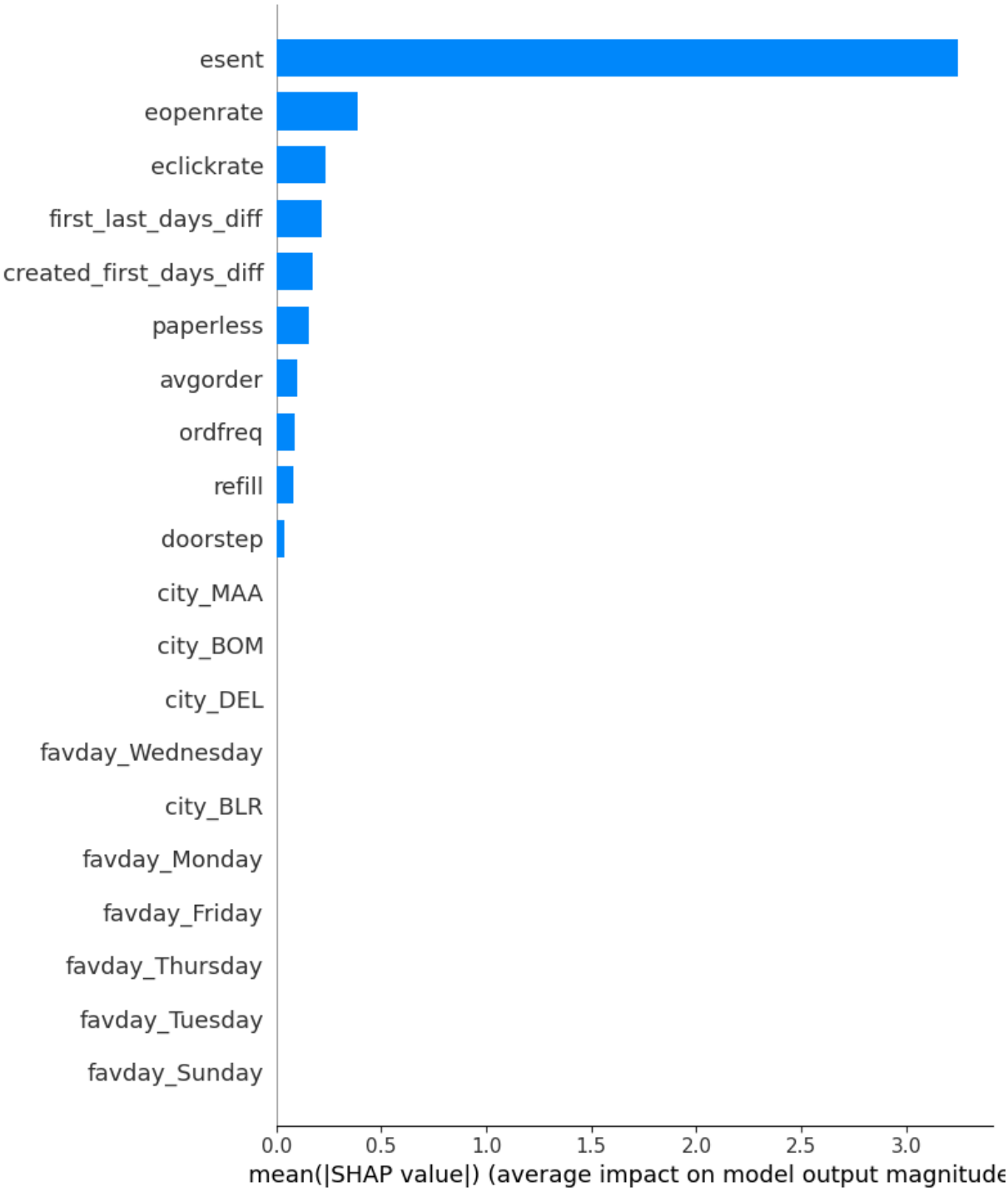
In [ ]:
```
plot_collection(trial, "average_shap")
```



Global Explanations

In [ ]:
```
shap_values = trial.tensor("full_shap/f0").value(trial.last_complete_step)
shap_no_base = shap_values[:, :-1]
```

```
shap_base_value = shap_values[0, -1]
shap.summary_plot(shap_no_base, plot_type="bar", feature_names=feature_names)
```

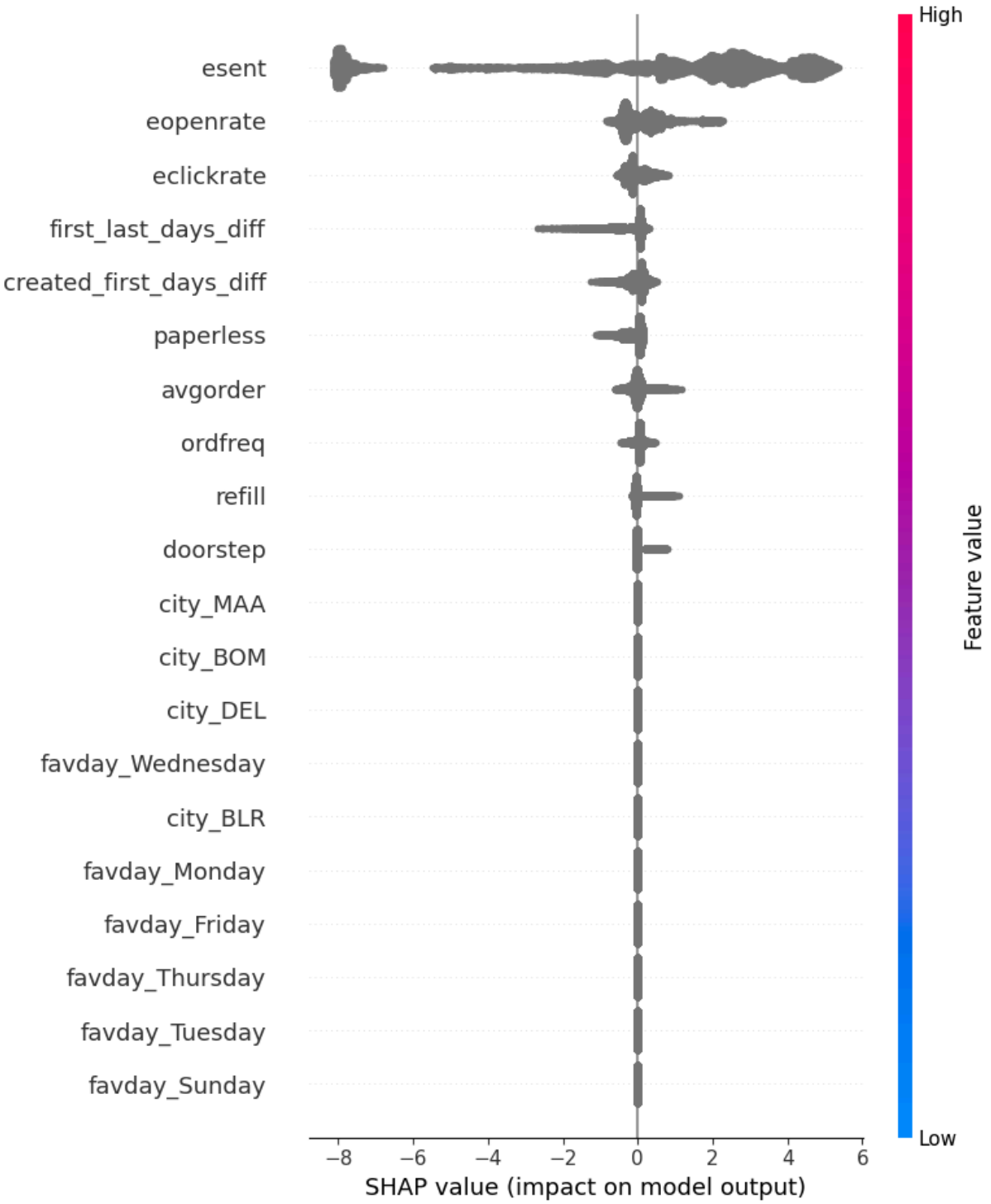


In [ ]:
```
shap_base_value
```

Out[ ]: 2.0374506

In [ ]:
```
train_shap = pd.DataFrame(train[:,1:],columns=feature_names)
```

In [ ]:
```
shap.summary_plot(shap_no_base, train_shap)
```



Local Explanations

In [ ]:
```
shap.initjs()
```

In [ ]:
```
shap.force_plot(
    shap_base_value,
    shap_no_base[100, :],
    train_shap.iloc[100, :],
```

```
    link="logit",
    matplotlib=False,
)
```

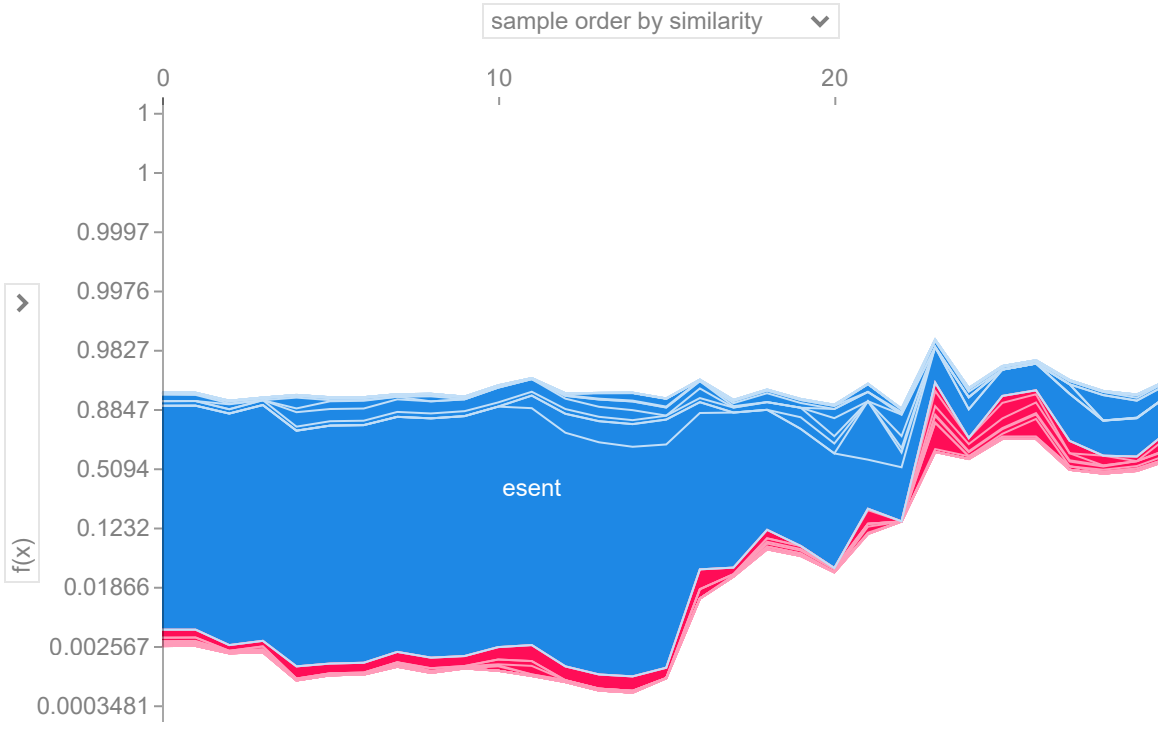
Out[ ]:

fir

In [ ]:
```
N_ROWS = shap_no_base.shape[0]
N_SAMPLES = min(100, N_ROWS)
sampled_indices = np.random.randint(N_ROWS, size=N_SAMPLES)
```

In [ ]:
```
shap.force_plot(
    shap_base_value,
    shap_no_base[sampled_indices, :],
    train_shap.iloc[sampled_indices, :],
    link="logit",
)
```

Out[ ]:



In [ ]: