```python
# Import packages
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

# Load data
df = pd.read_csv('/Users/rajaallmdar/Desktop/BCG-Data-Science-and-Analytics-Virtual-Job-Simulation/Data/clean_data_after_eda.csv')
df["date_activ"] = pd.to_datetime(df["date_activ"], format='%Y-%m-%d')
df["date_end"] = pd.to_datetime(df["date_end"], format='%Y-%m-%d')
df["date_modif_prod"] = pd.to_datetime(df["date_modif_prod"], format='%Y-%m-%d')
df["date_renewal"] = pd.to_datetime(df["date_renewal"], format='%Y-%m-%d')

# Handling missing values
df.fillna(df.mean(), inplace=True)  # numeric columns
df.fillna(df.mode().iloc[0], inplace=True)  # categorical columns

# Feature engineering

# Load price data
price_df = pd.read_csv('/Users/rajaallmdar/Desktop/BCG-Data-Science-and-Analytics-Virtual-Job-Simulation/Data/price_data.csv')
price_df["price_date"] = pd.to_datetime(price_df["price_date"], format='%Y-%m-%d')

# Group off-peak prices by companies and month
monthly_price_by_id = price_df.groupby(['id', 'price_date']).agg({'price_off_peak_var': 'mean', 'price_off_peak_fix': 'mean'}).reset_index()

# Get January and December prices
jan_prices = monthly_price_by_id.groupby('id').first().reset_index()
dec_prices = monthly_price_by_id.groupby('id').last().reset_index()

# Calculate the difference between off-peak prices in December and preceding January
diff = pd.merge(dec_prices.rename(columns={'price_off_peak_var': 'dec_1', 'price_off_peak_fix': 'dec_2'}), jan_prices.drop(columns='price_date'), on='id')
diff['offpeak_diff_dec_january_energy'] = diff['dec_1'] - diff['price_off_peak_var']
diff['offpeak_diff_dec_january_power'] = diff['dec_2'] - diff['price_off_peak_fix']
diff = diff[['id', 'offpeak_diff_dec_january_energy', 'offpeak_diff_dec_january_power']]

# Merge the feature with the main dataframe
df = pd.merge(df, diff, on='id', how='left')

# Additional feature engineering

# Calculate rolling average price over the previous 3 months
monthly_price_by_id['rolling_avg_3m'] = monthly_price_by_id.groupby('id')['price_off_peak_var'].transform(lambda x: x.rolling(3).mean())

# Calculate price volatility over the previous 3 months
monthly_price_by_id['volatility_3m'] = monthly_price_by_id.groupby('id')['price_off_peak_var'].transform(lambda x: x.rolling(3).std())

# Calculate trend over time
def calculate_trend(group):
    x = np.array(range(len(group))).reshape(-1, 1)
    y = group.values
    model = LinearRegression().fit(x, y)
    return model.coef_[0]

monthly_price_by_id['trend'] = monthly_price_by_id.groupby('id')['price_off_peak_var'].transform(calculate_trend)

# Encode the month
monthly_price_by_id['month'] = monthly_price_by_id['price_date'].dt.month

# Merge the additional features with the main dataframe
df = pd.merge(df, monthly_price_by_id, on='id', how='left')

# Prepare the data for modeling

# Select relevant features and target variable
features = ['offpeak_diff_dec_january_energy', 'offpeak_diff_dec_january_power', 'rolling_avg_3m', 'volatility_3m', 'trend', 'month']
target = 'churn'

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)

# Train the random forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
report = classification_report(y_test, y_pred)
print("Classification Report:\n", report)

# Investigate the discount strategy

# Load client data
client_data = pd.read_csv('/Users/rajaallmdar/Desktop/BCG-Data-Science-and-Analytics-Virtual-Job-Simulation/Data/client_data.csv')

# Merge the client data with the main dataframe
df = pd.merge(df, client_data, on='id', how='left')

# Select customers likely to churn
churn_customers = df[df['churn'] == 1]

# Apply the discount strategy (20% discount)
churn_customers['discounted_price_off_peak'] = churn_customers['price_off_peak_var'] * 0.8

# Print the discounted prices
print("Discounted Prices for Churn Customers:\n", churn_customers[['id', 'discounted_price_off_peak']])

# Save the updated dataframe
df.to_csv('/Users/rajaallmdar/Desktop/BCG-Data-Science-and-Analytics-Virtual-Job-Simulation/Data/processed_data.csv', index=False)
```