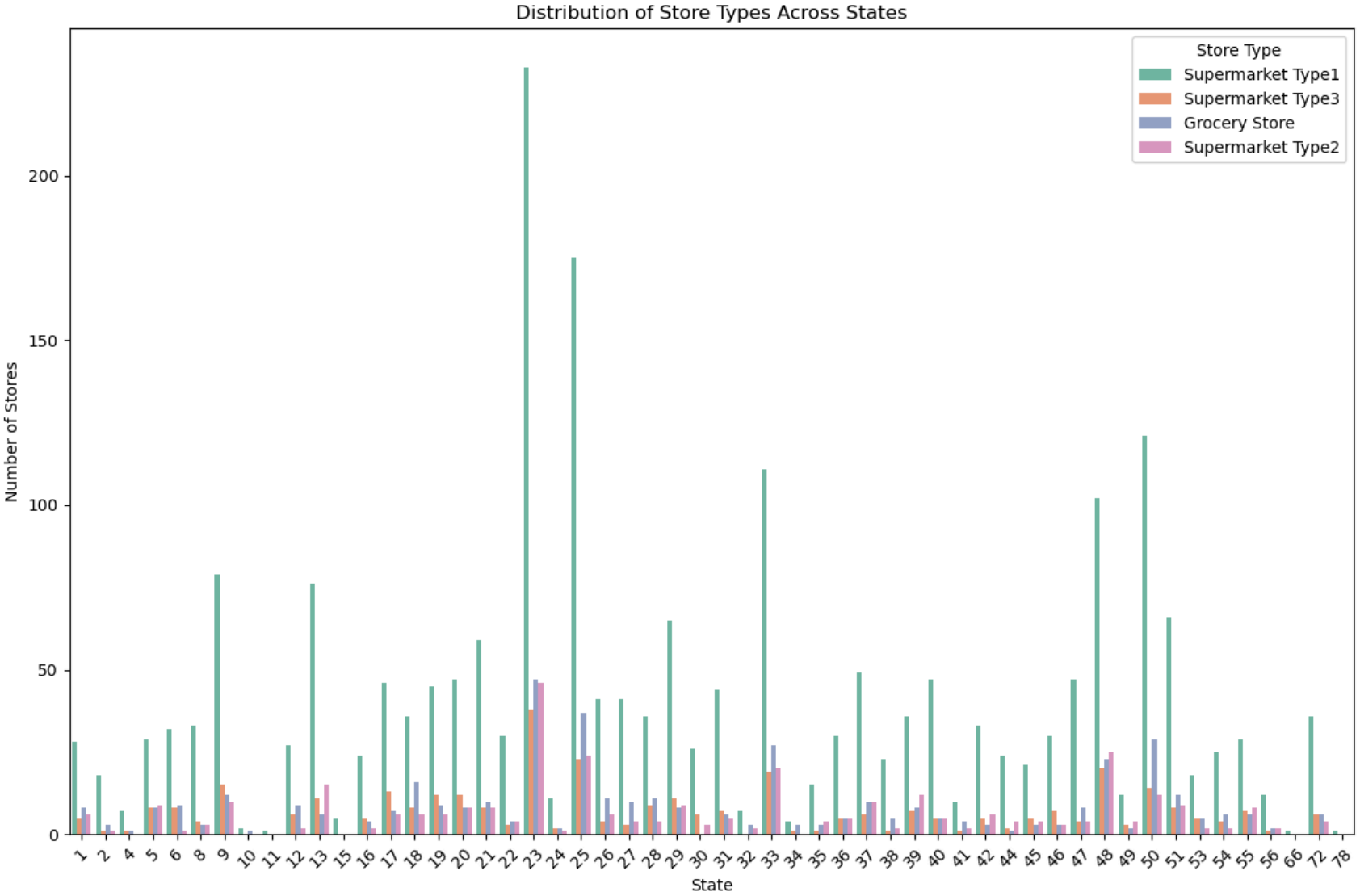
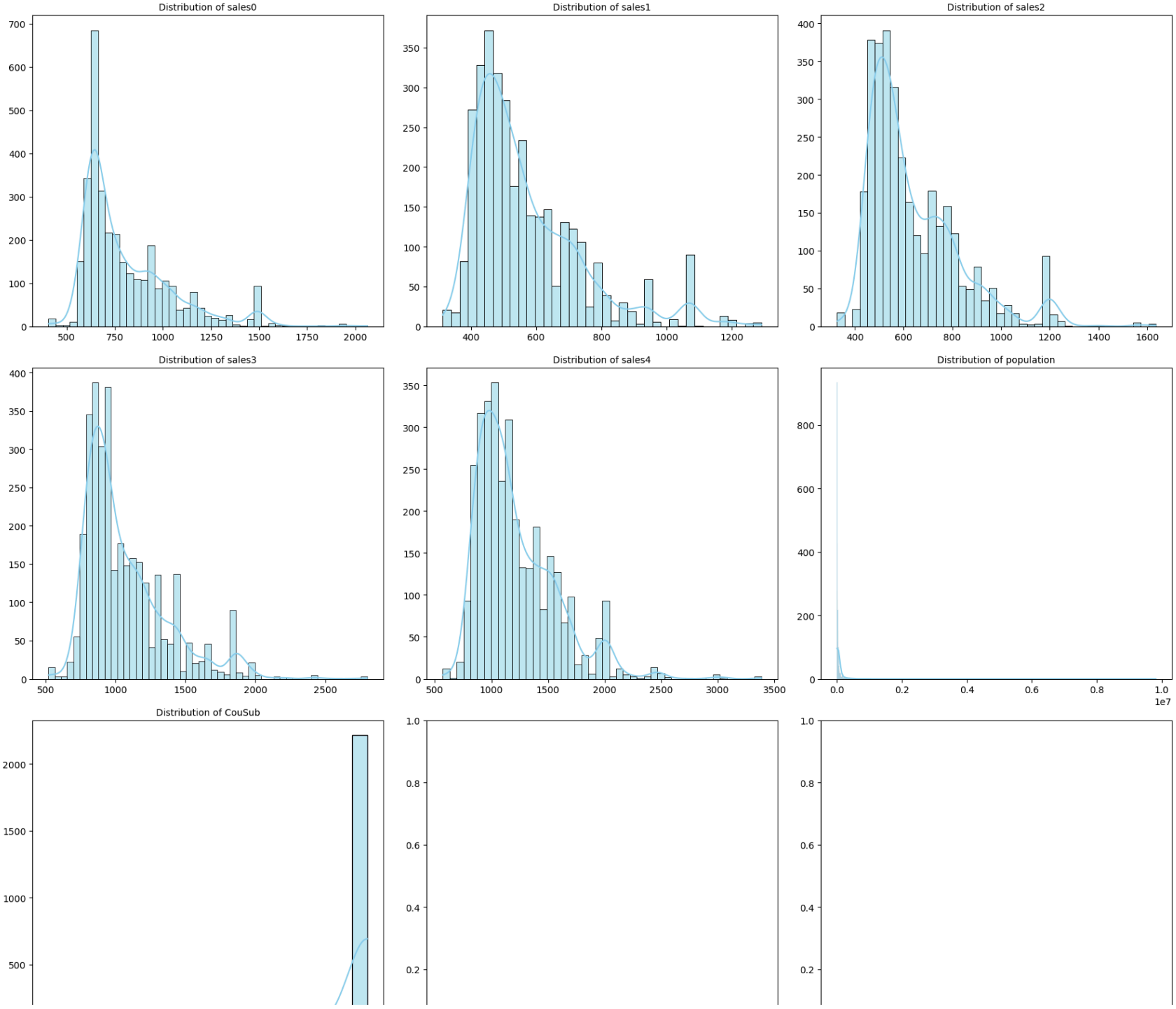
In [2]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve, auc
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns

# Load data with error handling for encoding issues
try:
    st_train = pd.read_csv('store_train.csv', encoding='utf-8')
    st_test = pd.read_csv('store_test.csv', encoding='utf-8')
except UnicodeDecodeError:
    st_train = pd.read_csv('store_train.csv', encoding='ISO-8859-1')
    st_test = pd.read_csv('store_test.csv', encoding='ISO-8859-1')

# Data preprocessing
st_train['store'] = np.where(st_train['store'] == 1, "Yes", "No")
st_train['store'] = LabelEncoder().fit_transform(st_train['store'] == "Yes")

# EDA: Distribution of 'store_Type' for each 'State'
plt.figure(figsize=(12, 8))
sns.countplot(data=st_train, x='State', hue='store_Type', palette='Set2')
plt.title('Distribution of Store Types Across States')
plt.xticks(rotation=45)
plt.xlabel('State')
plt.ylabel('Number of Stores')
plt.legend(title='Store Type')
plt.tight_layout()
plt.show()

# Adjusting layout and figure size for histograms of numeric data
numeric_cols = ['sales0', 'sales1', 'sales2', 'sales3', 'sales4', 'population', 'CouSub']
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(18, 16))  # Adjusted layout for 9 plots
axes = axes.flatten()  # Flatten the array of axes
for idx, col in enumerate(numeric_cols):
```

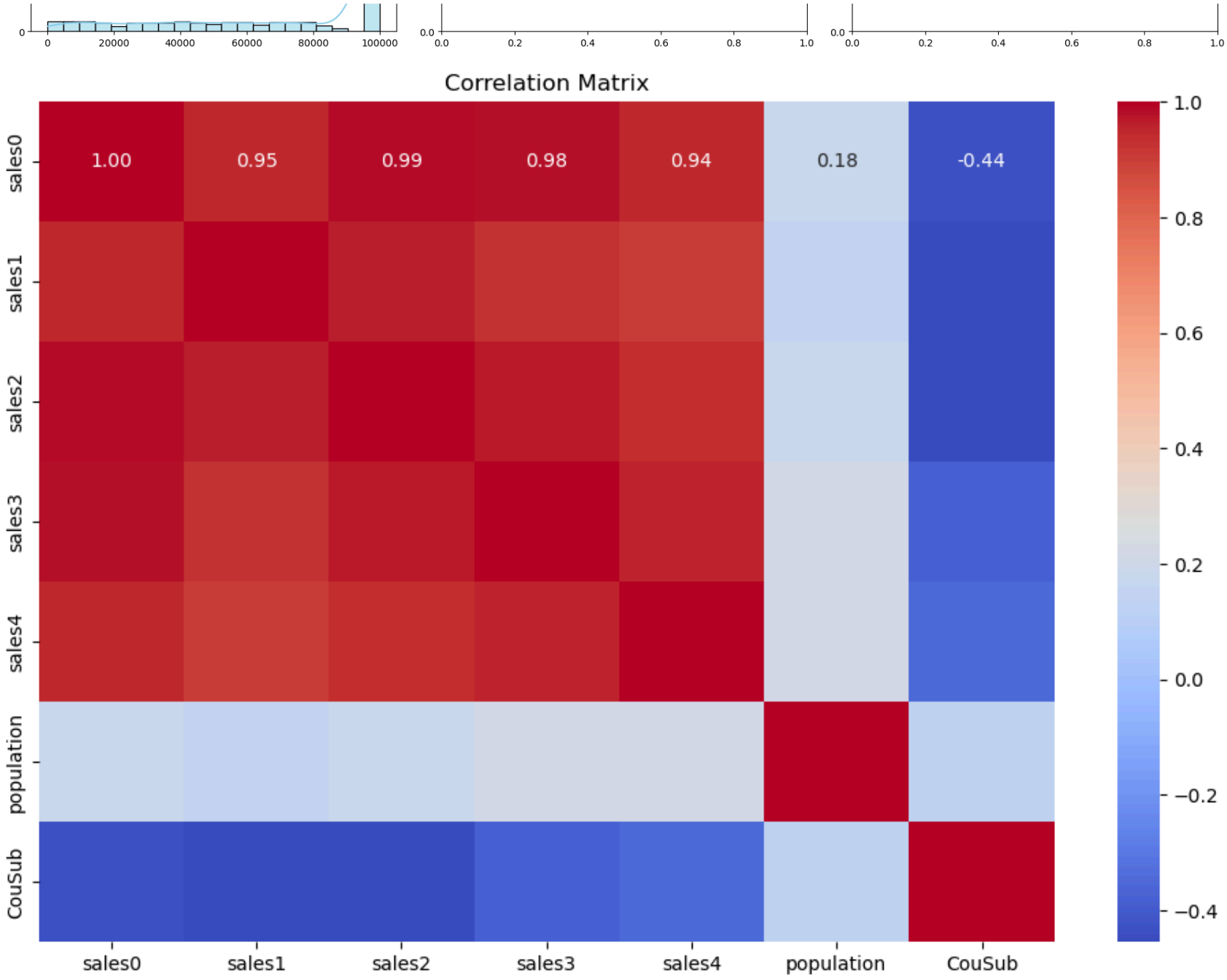
```
    if idx < len(numeric_cols):
        sns.histplot(st_train[col], ax=axes[idx], kde=True, color='skyblue')
        axes[idx].set_title(f'Distribution of {col}', fontsize=10)
        axes[idx].set_xlabel('')
        axes[idx].set_ylabel('')
plt.tight_layout()
plt.show()

# Correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(st_train[numeric_cols].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Define preprocessing pipeline
categorical_cols = ['country', 'State', 'countyname', 'storecode', 'Areaname', 'countytownname', 'state_alpha', 'stor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='median'), numeric_cols),
        ('cat', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_cols)
    ])

# Random forest model and pipeline
rf = RandomForestClassifier(random_state=42)
pipe = Pipeline(steps=[('preprocessor', preprocessor),
                       ('classifier', rf)])

# Setup grid search
param_grid = {
    'classifier__n_estimators': [100, 500],
    'classifier__max_features': ['auto', 'sqrt'],
    'classifier__min_samples_split': [10, 20]
}
grid_search = GridSearchCV(pipe, param_grid, cv=5, verbose=3, scoring='roc_auc', n_jobs=-1)
grid_search.fit(st_train.drop('store', axis=1), st_train['store'])

# Model Evaluation: ROC Curve
y_test = st_train['store']
y_pred_prob = grid_search.predict_proba(st_train.drop('store', axis=1))[:, 1]
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# Feature Importance
importances = grid_search.best_estimator_.named_steps['classifier'].feature_importances_
features = st_train.drop('store', axis=1).columns  # This gets all the feature names directly from the DataFrame

# Sorting indices of features based on their importance
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(12, 6))
plt.title('Feature Importances')
plt.barh(range(len(importances)), importances[indices], color='b', align='center')  # Corrected to plot all importanc
plt.yticks(range(len(importances)), [features[i] for i in indices])  # Corrected to match indices with feature names
plt.xlabel('Relative Importance')
plt.gca().invert_yaxis()  # Invert y-axis to have the most important feature at the top
plt.show()
```


Distribution of Store Types Across States
Store Type
Supermarket Type1
Supermarket Type3
Grocery Store
Supermarket Type2
Number of Stores
0
50
100
150
200
State
1 2 4 5 6 8 9 10 11 12 13 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 44 45 46 47 48 49 50 51 53 54 55 56 66 72 78

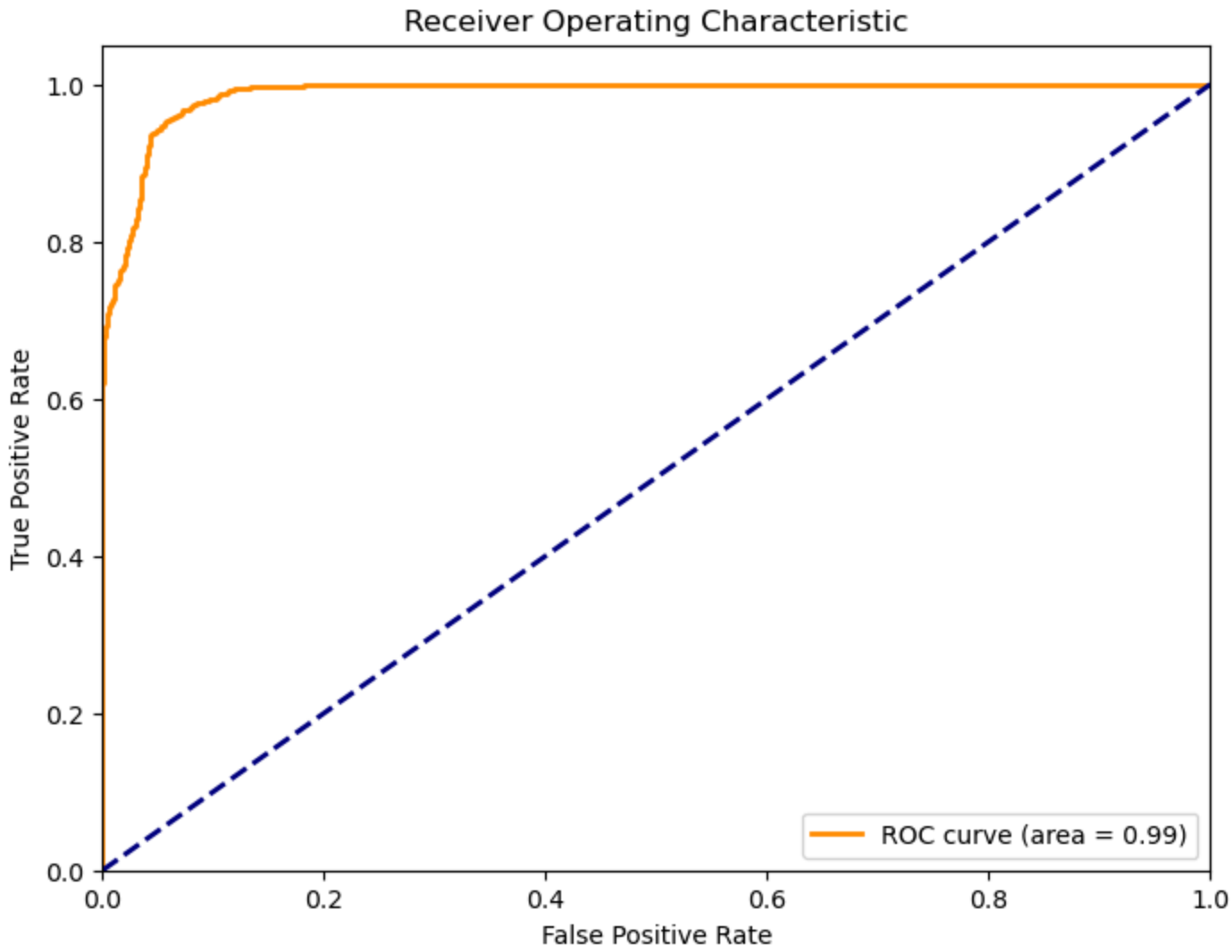
```
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\vpark\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Distribution of sales0
Distribution of sales1
Distribution of sales2
Distribution of sales3
Distribution of sales4
Distribution of population
1e7
Distribution of CouSub

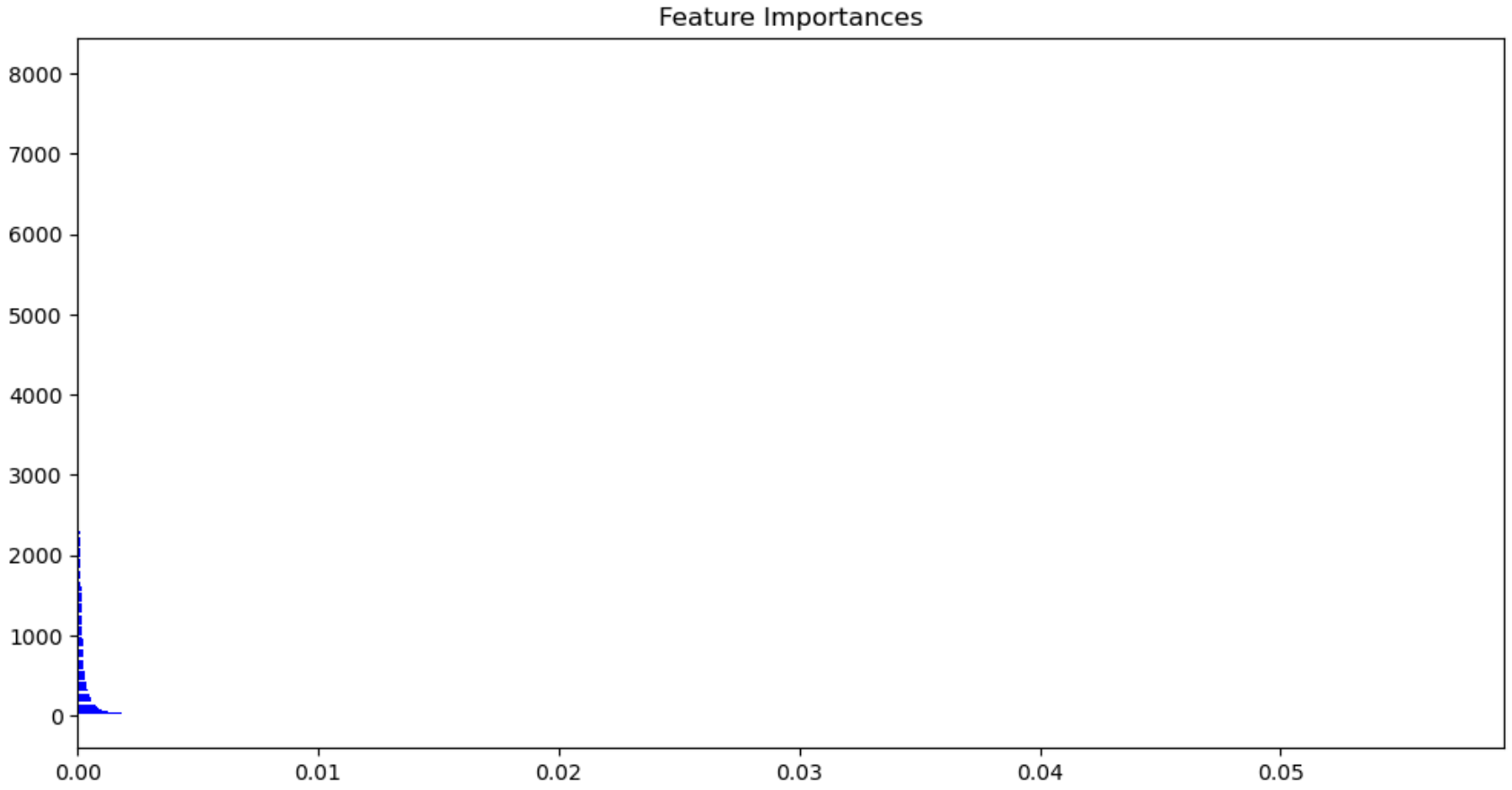



```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
```

```
C:\Users\vpark\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:424: FutureWarning: `max_features='auto'` has
been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'` o
r remove this parameter as it is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```



```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[2], line 107
    105 plt.title('Feature Importances')
    106 plt.barh(range(len(importances)), importances[indices], color='b', align='center')  # Corrected to plot all i
mportances
--> 107 plt.yticks(range(len(importances)), [features[i] for i in indices])  # Corrected to match indices with featur
e names
    108 plt.xlabel('Relative Importance')
    109 plt.gca().invert_yaxis()  # Invert y-axis to have the most important feature at the top

Cell In[2], line 107, in <listcomp>(.0)
    105 plt.title('Feature Importances')
    106 plt.barh(range(len(importances)), importances[indices], color='b', align='center')  # Corrected to plot all i
mportances
--> 107 plt.yticks(range(len(importances)), [features[i] for i in indices])  # Corrected to match indices with featur
e names
    108 plt.xlabel('Relative Importance')
    109 plt.gca().invert_yaxis()  # Invert y-axis to have the most important feature at the top

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:5366, in Index.__getitem__(self, key)
   5363 if is_integer(key) or is_float(key):
   5364     # GH#44051 exclude bool, which would return a 2d ndarray
   5365     key = com.cast_scalar_indexer(key)
-> 5366     return getitem(key)
   5368 if isinstance(key, slice):
   5369     # This case is separated from the conditional above to avoid
   5370     # pessimization com.is_bool_indexer and ndim checks.
   5371     return self._getitem_slice(key)

IndexError: index 302 is out of bounds for axis 0 with size 16
```


Feature Importances
8000
7000
6000
5000
4000
3000
2000
1000
0
0.00
0.01
0.02
0.03
0.04
0.05

In [ ]: