capstoneproject2

April 5, 2025

[87]:
```
#import essential libraries
#!pip install pmdarima==2.0.3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from pmdarima import auto_arima
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,␣
↪mean_absolute_percentage_error
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy import stats
from sklearn.preprocessing import LabelEncoder
```

0.0.1 1. Data Cleaning and Preparation

[98]:
```
print("Loading and preparing data...")
file_path = 'Work_Permits_Data_ud.xlsx'
df = pd.read_excel(file_path)

# Convert to datetime index
df['Date'] = pd.to_datetime(df[['Year', 'Month']].assign(day=1))
df.set_index('Date', inplace=True)

# Handle duplicates
df = df[~df.index.duplicated(keep='first')]
df = df.asfreq('MS')  # Monthly start frequency

# Handle missing values
print("\nHandling missing values...")
print("Missing values before cleaning:")
print(df.isnull().sum())
numeric_cols = df.select_dtypes(include=[np.number]).columns
df[numeric_cols] = df[numeric_cols].ffill().bfill()
```

```
# Handle outliers only for numeric columns
print("Handling outliers...")
for col in numeric_cols:
    z_scores = np.abs(stats.zscore(df[col]))
    df = df[(z_scores < 3)]

# Final check
if df.isnull().any().any():
    print("Warning: NaN values still present - dropping remaining")
    df = df.dropna()

print("\nMissing values after cleaning:")
print(df.isnull().sum())

# Display cleaned data info
display(df.head())
display(df.describe())
```

```
Loading and preparing data…

Handling missing values…
Missing values before cleaning:
Year                              0
Month                             0
Country                           0
Number of Permits Issued          0
Number of Permits Expired         0
Permit Type                       0
Renewal Rate                      0
Institution Type                  0
Program Level                     0
New Eligibility Requirements     83
Acceptance Rate                   0
Unemployment Rate                 0
Job Vacancy Rate                  0
Policy Change Date                0
Impact Assessment                 0
dtype: int64
Handling outliers…
Warning: NaN values still present - dropping remaining

Missing values after cleaning:
Year                            0
Month                           0
Country                         0
Number of Permits Issued        0
Number of Permits Expired       0
```

```
Permit Type                     0
Renewal Rate                    0
Institution Type                0
Program Level                   0
New Eligibility Requirements    0
Acceptance Rate                 0
Unemployment Rate               0
Job Vacancy Rate                0
Policy Change Date              0
Impact Assessment               0
dtype: int64
```

Date	Year	Month	Country	Number of Permits Issued	Number of Permits Expired	Permit Type	Renewal Rate	Institution Type	Program Level	New Eligibility Requirements	Acceptance Rate	Unemployment Rate	Job Vacancy Rate	Policy Change Date	Impact Assessment
2010-03-01	2010	3	Australia	9491	4514	Permanent Residence	0.9621	Private	Diploma	Language Test Required	0.8429	0.0762	0.0319	2015-09-30	Positive
2010-05-01	2010	5	USA	543	2952	Study	0.8297	University	Diploma	Language Test Required	0.7986	0.0661	0.0637	2020-07-31	Neutral
2010-08-01	2010	8	India	2499	4061	Study	0.5163	Private	Diploma	Language Test Required	0.8320	0.1089	0.0535		
2010-09-01	2010	9	UK	8861	3348	Permanent Residence	0.9172	University	Diploma	Language Test Required	0.6952	0.0319	0.0253		
2010-10-01	2010	10	USA	3483	4665	Permanent Residence	0.6460	Private	Certificate	Language Test Required	0.9441	0.0879	0.0418		

```
2010-08-01          2023-11-30           Positive
2010-09-01          2022-12-31           Negative
2010-10-01          2022-07-31            Neutral
```

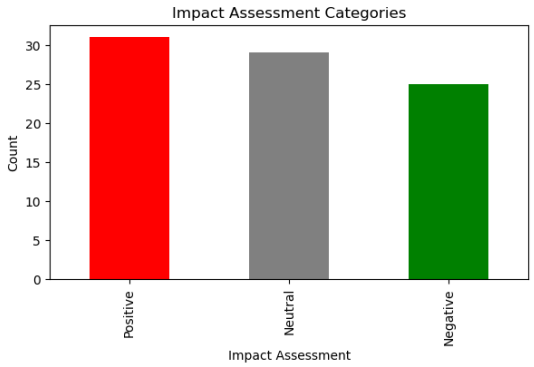
```
              Year       Month  Number of Permits Issued  \
count    85.000000   85.000000                 85.000000
mean   2016.141176    6.717647               5565.729412
min    2010.000000    1.000000                338.000000
25%    2014.000000    4.000000               2977.000000
50%    2016.000000    7.000000               6144.000000
75%    2019.000000    9.000000               8040.000000
max    2023.000000   12.000000               9950.000000
std       3.566220    3.379250               2835.858514

       Number of Permits Expired  Renewal Rate  Acceptance Rate  \
count                  85.000000     85.000000        85.000000
mean                 2701.658824      0.751596         0.762819
min                    62.000000      0.516300         0.601700
25%                  1567.000000      0.646000         0.689600
50%                  2855.000000      0.770100         0.754600
75%                  3935.000000      0.849300         0.837500
max                  4813.000000      0.975900         0.944100
std                  1441.899777      0.136168         0.094497

       Unemployment Rate  Job Vacancy Rate             Policy Change Date
count          85.000000         85.000000                             85
mean            0.088765          0.057951  2019-09-09 06:29:38.823529472
min             0.031900          0.020700            2015-03-31 00:00:00
25%             0.060200          0.038200            2017-08-31 00:00:00
50%             0.088600          0.056700            2019-07-31 00:00:00
75%             0.118300          0.079500            2021-12-31 00:00:00
max             0.149100          0.099900            2023-12-31 00:00:00
std             0.034497          0.023136                            NaN
```

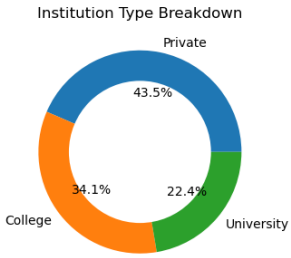
0.0.2 2. Exploratory Data Analysis (EDA)

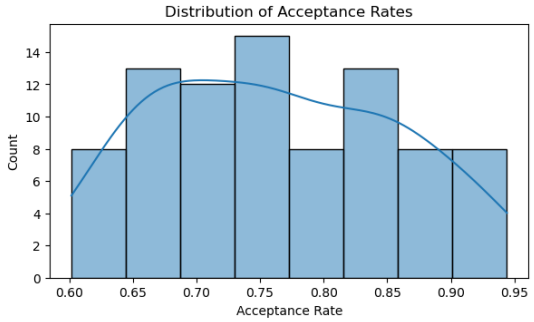
[106]:
```
print("\n=== Performing Enhanced EDA ===")
plt.figure(figsize=(15, 20))
plt.subplots_adjust(hspace=0.5)

# 1. Categorical Analysis - Impact Assessment
if 'Impact Assessment' in df.columns:
    plt.subplot(4, 2, 1)
    impact_counts = df['Impact Assessment'].value_counts()
    impact_counts.plot(kind='bar', color=['red', 'gray', 'green'])
    plt.title('Impact Assessment Categories')
    plt.ylabel('Count')
```

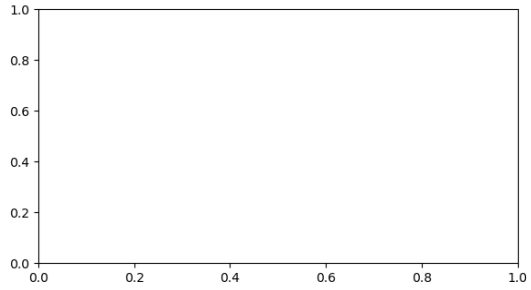
```
# 3. Donut chart for Institution Type
plt.subplot(4, 2, 2)
if 'Institution Type' in df.columns:
    inst_counts = df['Institution Type'].value_counts()
    plt.pie(inst_counts, labels=inst_counts.index, autopct='%1.1f%%')
    centre_circle =plt.Circle((0,0), 0.70, fc='white')
    fig = plt.gcf()
    fig.gca().add_artist(centre_circle)
    plt.title('Institution Type Breakdown')

# 4. Distribution Charts
plt.subplot(4, 2, 3)
if 'Acceptance Rate' in df.columns:
    sns.histplot(df['Acceptance Rate'], kde=True)
    plt.title('Distribution of Acceptance Rates')

# 5. Comparison Charts - Permits Issued vs Expired by Country
plt.subplot(4, 2, 4)
if 'Country' in df.columns and 'Number of Permits Expired' in df.columns:
    country_stats = df.groupby('Country')[['Number of Permits Issued', 'Number␣
 ↪of Permits Expired']].mean()
    country_stats.plot(kind='bar', width=0.8)
    plt.title('Permits Issued vs Expired by Country')
    plt.ylabel('Average Count')
    plt.xticks(rotation=45)

plt.show()
```

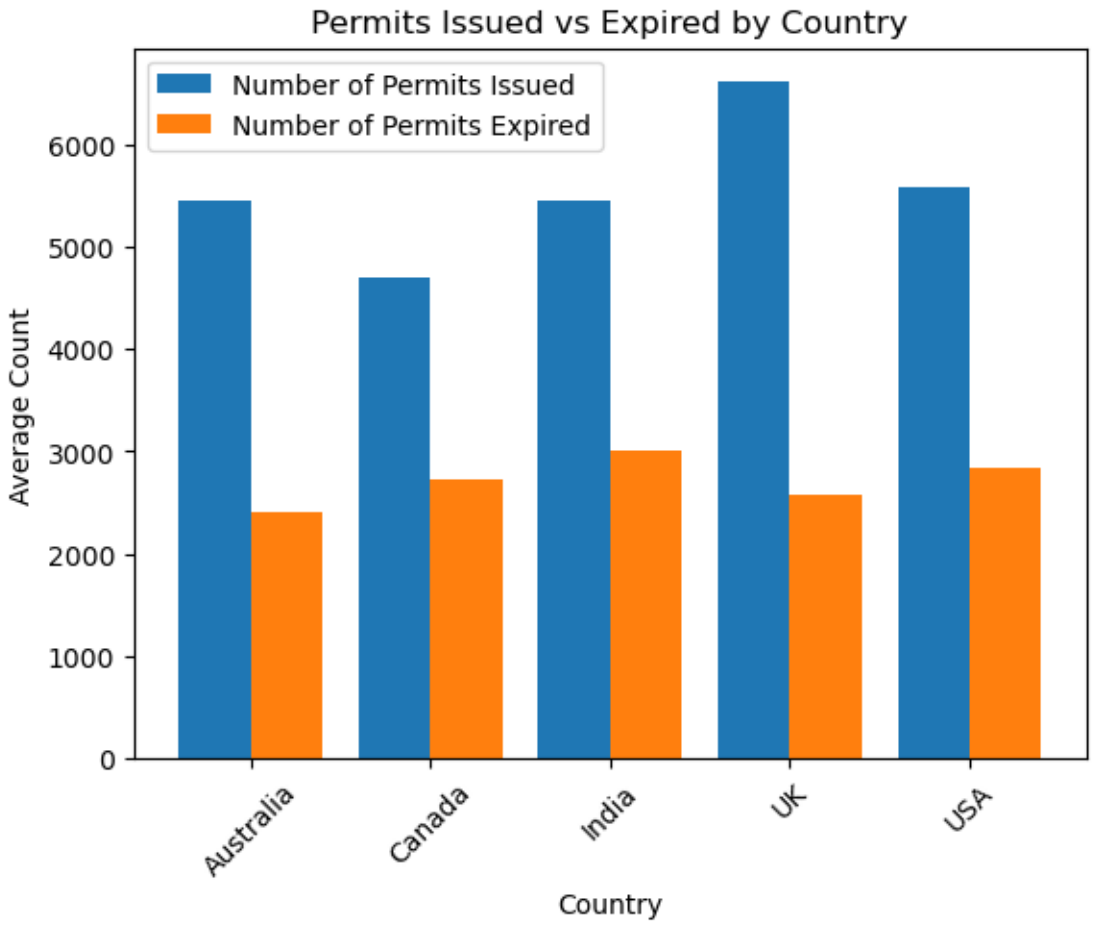
```
=== Performing Enhanced EDA ===
```


Impact Assessment Categories
Count
Positive
Neutral
Negative
Impact Assessment


Institution Type Breakdown
Private
43.5%
34.1%
22.4%
College
University


Distribution of Acceptance Rates
Count
Acceptance Rate





0.0.3 3. Feature Engineering for ML Models

[74]:
```
print("\nPreparing features for machine learning...")
ml_data = df.copy()

# Extract datetime features before encoding
ml_data['Year'] = ml_data.index.year
ml_data['Month'] = ml_data.index.month
ml_data['Day'] = ml_data.index.day

# Encode categorical variables
cat_cols = ml_data.select_dtypes(include=['object', 'category']).columns
for col in cat_cols:
    le = LabelEncoder()
    ml_data[col] = le.fit_transform(ml_data[col].astype(str))

# Ensure all features are numeric
ml_data = ml_data.select_dtypes(include=[np.number])
```

```
Preparing features for machine learning...
```

0.0.4 4. Predictive Modeling

[78]:
```
print("\n=== Building Predictive Models ===")
target = 'Number of Permits Issued'
train_size = int(len(ml_data) * 0.8)

# Split features and target
X = ml_data.drop(columns=[target])
y = ml_data[target]
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Random Forest
print("\nTraining Random Forest...")
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_preds = rf.predict(X_test)
rf_mae = mean_absolute_error(y_test, rf_preds)
print(f"Random Forest MAE: {rf_mae:.2f}")

# XGBoost
print("\nTraining XGBoost...")
xgb_model = xgb.XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)
xgb_preds = xgb_model.predict(X_test)
xgb_mae = mean_absolute_error(y_test, xgb_preds)
print(f"XGBoost MAE: {xgb_mae:.2f}")
```

```
# SARIMA (uses only the target variable)
print("\nTraining SARIMA model...")
train_ts, test_ts = y[:train_size], y[train_size:]

try:
    sarima_model = auto_arima(train_ts, seasonal=True, m=12,
                              trace=True, suppress_warnings=True)
    print(f"Best SARIMA order: {sarima_model.order}")

    sarima = ARIMA(train_ts, order=sarima_model.order,
                   seasonal_order=(1,1,1,12)).fit()
    sarima_preds = sarima.get_forecast(len(test_ts)).predicted_mean
    sarima_mae = mean_absolute_error(test_ts, sarima_preds)
    print(f"SARIMA MAE: {sarima_mae:.2f}")

    # 5-year forecast
    future = sarima.get_forecast(60).predicted_mean
    future_dates = pd.date_range(start=df.index[-1] + pd.offsets.MonthBegin(1),
                                 periods=60, freq='MS')

    # Plot 1: Historical Data Only
    plt.figure(figsize=(14,6))
    plt.plot(df.index, df[target], color='blue', label='Historical Data')
    plt.title('Historical Work Permits Issued')
    plt.xlabel('Year')
    plt.ylabel('Number of Permits')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Plot 2: 5-Year Forecast Only
    plt.figure(figsize=(14,6))
    plt.plot(future_dates, future, color='red', linestyle='--', label='5-Year␣
 ↪Forecast')
    plt.title('5-Year Work Permits Forecast (SARIMA)')
    plt.xlabel('Year')
    plt.ylabel('Number of Permits')
    plt.legend()
    plt.grid(True)
    plt.show()

except Exception as e:
    print(f"SARIMA failed: {e}")
    sarima_mae = np.nan
```

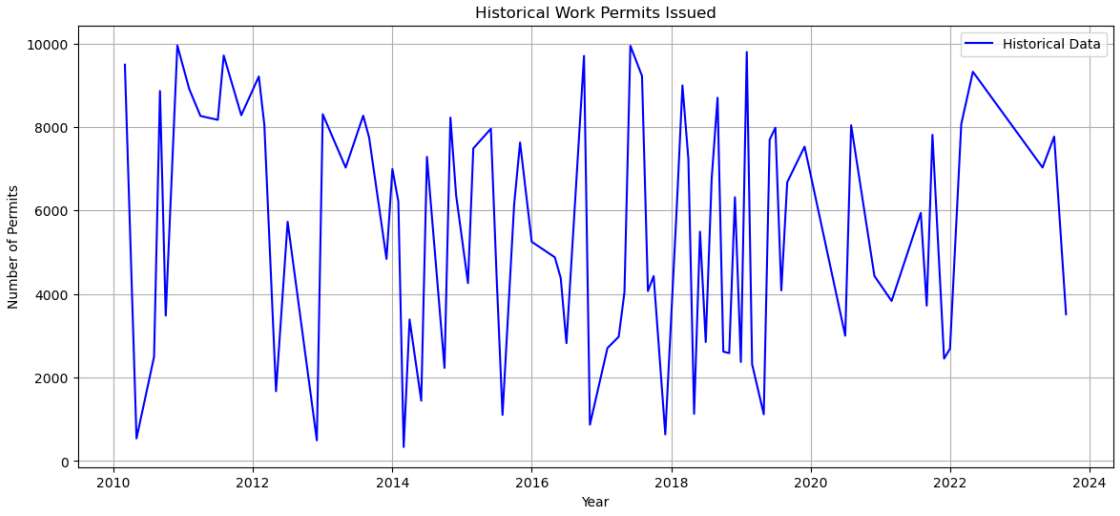
```
=== Building Predictive Models ===
```

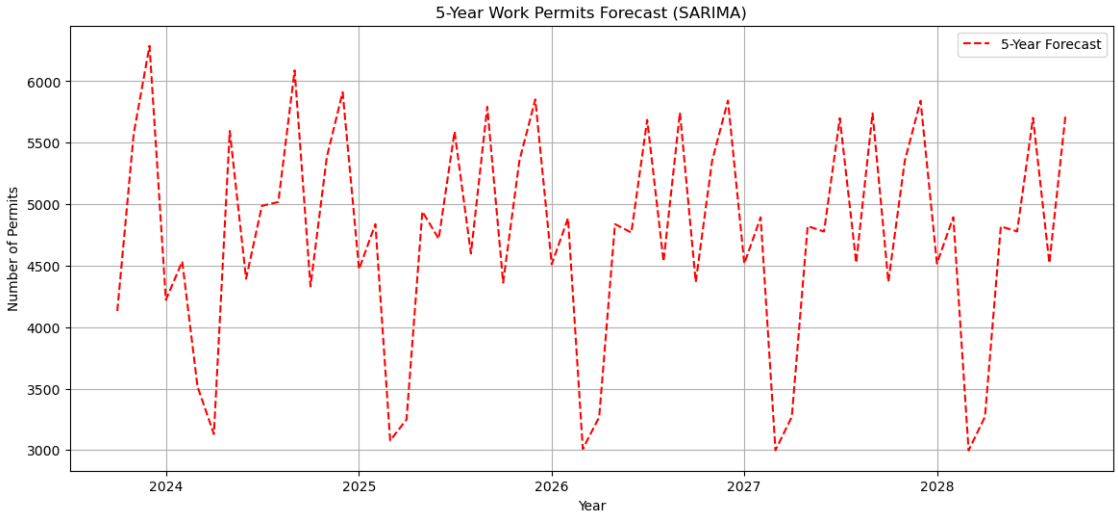
```
Training Random Forest…
Random Forest MAE: 2351.48

Training XGBoost…
XGBoost MAE: 2373.66

Training SARIMA model…
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(1,0,1)[12] intercept   : AIC=inf, Time=0.34 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=1283.711, Time=0.01 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=1287.466, Time=0.03 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=1287.107, Time=0.03 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=1384.395, Time=0.00 sec
 ARIMA(0,0,0)(1,0,0)[12] intercept   : AIC=1285.646, Time=0.02 sec
 ARIMA(0,0,0)(0,0,1)[12] intercept   : AIC=1285.135, Time=0.02 sec
 ARIMA(0,0,0)(1,0,1)[12] intercept   : AIC=1287.135, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[12] intercept   : AIC=1285.691, Time=0.01 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=1285.682, Time=0.03 sec
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=1287.763, Time=0.02 sec

Best model:  ARIMA(0,0,0)(0,0,0)[12] intercept
Total fit time: 0.553 seconds
Best SARIMA order: (0, 0, 0)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has
been provided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has
been provided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has
been provided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-invertible
starting seasonal moving average Using zeros as starting parameters.
  warn('Non-invertible starting seasonal moving average'
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index
is available. Prediction results will be given with an integer index beginning
at `start`.
  return get_prediction_index(
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported
index is available. In the next version, calling this method in a model without
a supported index will result in an exception.
  return get_prediction_index(
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index
is available. Prediction results will be given with an integer index beginning
at `start`.
  return get_prediction_index(
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported
index is available. In the next version, calling this method in a model without
a supported index will result in an exception.
  return get_prediction_index(
```

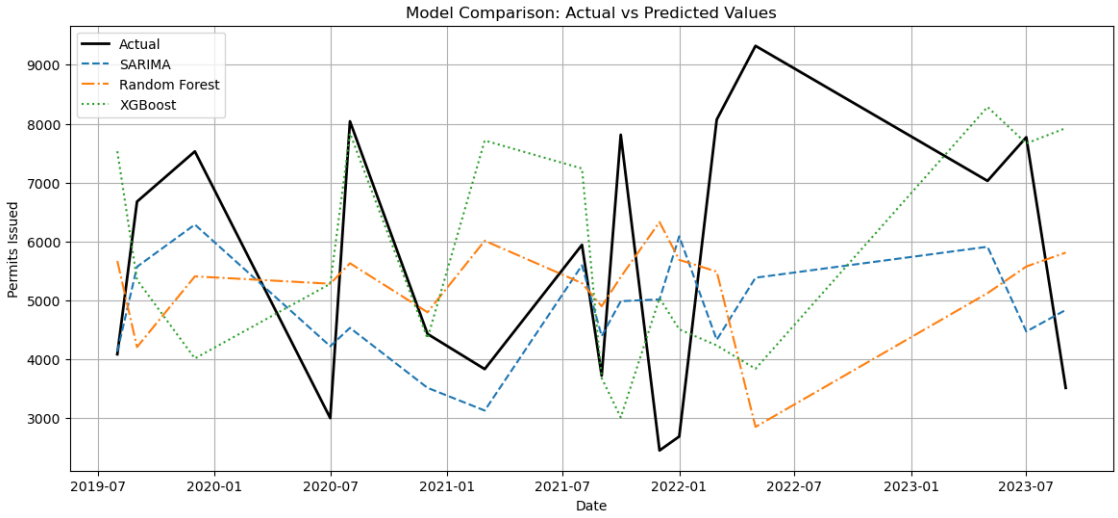
```
SARIMA MAE: 1879.89
```





0.0.5 5. Model Comparison

[81]:
```
# Model comparison plot
plt.figure(figsize=(14,6))
plt.plot(y_test.index, y_test, label='Actual', color='black', linewidth=2)
if not np.isnan(sarima_mae):
    plt.plot(test_ts.index, sarima_preds, label='SARIMA', linestyle='--')
plt.plot(y_test.index, rf_preds, label='Random Forest', linestyle='-.')
plt.plot(y_test.index, xgb_preds, label='XGBoost', linestyle=':')
plt.title('Model Comparison: Actual vs Predicted Values')
plt.xlabel('Date')
plt.ylabel('Permits Issued')
plt.legend()
plt.grid(True)
plt.show()
```



0.0.6 6. Results Summary

[84]:
```
print("\n=== Final Results ===")
results = {
    'SARIMA': sarima_mae,
    'Random Forest': rf_mae,
    'XGBoost': xgb_mae
}

valid_results = {k: v for k, v in results.items() if not np.isnan(v)}

if valid_results:
    best_model = min(valid_results, key=valid_results.get)
    best_mae = valid_results[best_model]

    print("\nAnswer to Research Question 1:")
    print(f"Work permits can be forecast with MAE of {best_mae:.2f}")

    print("\nAnswer to Research Question 2:")
    print(f"The best model is {best_model} with MAE of {best_mae:.2f}")

    print("\nModel Performance:")
    for model, mae in valid_results.items():
        print(f"- {model}: MAE = {mae:.2f}")
else:
    print("\nNo models were successfully trained")
```

```
=== Final Results ===
```

```
Answer to Research Question 1:
Work permits can be forecast with MAE of 1879.89

Answer to Research Question 2:
The best model is SARIMA with MAE of 1879.89

Model Performance:
- SARIMA: MAE = 1879.89
- Random Forest: MAE = 2351.48
- XGBoost: MAE = 2373.66
```