Forecasting App

Release Alpha - 1.1

Yerko E. Veliz Olivares

Jun 18, 2024

CONTENTS:

CHAPTER

ONE

S+

1.1 Adf_test module

Adf_test.**perform_adf_test_with_differencing**(*data*, *max_differencing=2*)

Perform ADF test with differencing up to a maximum number of differencing steps.

Args:
data (pd.DataFrame): Data for the ADF test. max_differencing (int): Maximum number of differencing steps.

Returns:
pd.DataFrame: ADF test results.

1.2 Arima module

Arima.**forecast_future**(*arima_results*, *df*, *start_year*, *forecast_until_year=2100*, *replace_negative_forecast=False*)

Forecast future values using ARIMA models.

Args:
arima_results (dict): ARIMA results. df (pd.DataFrame): Data frame containing the data. start_year (int): Start year for the forecast. forecast_until_year (int): Year until which to forecast. replace_negative_forecast (bool): Whether to replace negative forecast values with zero.

Returns:
dict: Forecast results for each country.

Arima.**optimize_arima**(*series*, *p_range*, *d_range*, *q_range*)

Optimize ARIMA model parameters.

Args:
series (pd.Series): Time series data. p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values.

Returns:
tuple: Best AIC, best order, best model.

Arima.**optimize_arima_models**(*adf_results*, *df*, *selected_countries*, *p_range*, *d_range*, *q_range*, *start_year*, *end_year*)

Optimize ARIMA models for multiple countries.

Args:

adf_results (pd.DataFrame): ADF test results. df (pd.DataFrame): Data frame containing the data. selected_countries (list): List of selected countries. p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values. start_year (int): Start year for the data. end_year (int): End year for the data.

Returns:

dict: ARIMA results for each country.

1.3 Arimax module

Arimax.**forecast_future**(*arimax_results*, *df*, *start_year*, *forecast_until_year=2100*, *replace_negative_forecast=False*)

Forecast future values using ARIMAX models.

Args:

arimax_results (dict): ARIMAX results. df (pd.DataFrame): Data frame containing the data. start_year (int): Start year for the forecast. forecast_until_year (int): Year until which to forecast. replace_negative_forecast (bool): Whether to replace negative forecast values with zero.

Returns:

dict: Forecast results for each country.

Arimax.**optimize_arimax**(*series*, *p_range*, *d_range*, *q_range*, *exog_series*)

Optimize ARIMAX model parameters.

Args:

series (pd.Series): Time series data. p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values. exog_series (pd.DataFrame): Exogenous variables.

Returns:

tuple: Best AIC, best order, best model.

Arimax.**optimize_arimax_models**(*adf_results*, *df*, *selected_countries*, *p_range*, *d_range*, *q_range*, *start_year*, *end_year*)

Optimize ARIMAX models for multiple countries.

Args:

adf_results (pd.DataFrame): ADF test results. df (pd.DataFrame): Data frame containing the data. selected_countries (list): List of selected countries. p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values. start_year (int): Start year for the data. end_year (int): End year for the data.

Returns:

dict: ARIMAX results for each country.

1.4 GroupPanel module

class GroupPanel.**GroupPanelWindow**(*main_window*, *is_forecast=False*)

Bases: QWidget

Group panel window for creating groups.

accept()

Accept the group creation.

cancel()

Cancel the group creation.

1.5 Mainwindow module

class Mainwindow.**MainWindow**

Bases: QMainWindow

Main application window for forecasting.

add_line(*name*, *value*, *color*, *line_type*)

Add a new line to the active lines list.

Args:

name (str): Name of the line. value (float): Value at which the line will be drawn. color (str): Color of the line. line_type (str): Type of the line (solid, dashed, dotted).

apply_forecast_corrections()

Apply corrections to the forecasted data.

apply_linear_correction(*country*, *target_year*, *target_value*, *variable*, *continuous*, *short*, *start*)

Apply linear correction to the forecasted data.

Args:

country (str): The country for which to apply the correction. target_year (int): The target year for the correction. target_value (float): The target value for the correction. variable (str): The variable to correct. continuous (bool): Whether to apply continuous correction. short (bool): Whether to apply short correction. start (bool): Whether to apply start correction.

apply_plot_settings(*x_range*, *y_range*, *title*, *legend_size*, *xlabel*, *ylabel*, *xlabel_size*, *ylabel_size*)

Apply plot settings to the current plot.

Args:

x_range (tuple): Range for the x-axis. y_range (tuple): Range for the y-axis. title (str): Title of the plot. legend_size (int): Font size for the legend. xlabel (str): Label for the x-axis. ylabel (str): Label for the y-axis. xlabel_size (int): Font size for the x-axis label. ylabel_size (int): Font size for the y-axis label.

clear_all()

Clear all selected forecasted models.

closeEvent(*event*)

Handle the event of closing the side panel when closing the main window.

Args:

event (QCloseEvent): The close event.

`create_group`(*group_name*)

Create a group with the specified name from selected countries.

Args:
group_name (str): The name of the group.

`download_plot`()

Download the current plot as an image.

`filter_country_list`()

Filter the list of countries based on the search input.

`filter_forecasted_country_list`()

Filter the list of forecasted countries based on the search input.

`format_adf_results`(*adf_results*)

Format ADF test results as HTML.

Args:
adf_results (pd.DataFrame): The ADF test results.

Returns:
str: The formatted results in HTML.

`format_arima_results`(*arima_results*)

Format ARIMA results as HTML.

Args:
arima_results (dict): The ARIMA results.

Returns:
str: The formatted results in HTML.

`format_arimax_results`(*arimax_results*)

Format ARIMAX results as HTML.

Args:
arimax_results (dict): The ARIMAX results.

Returns:
str: The formatted results in HTML.

`format_sarimax_results`(*sarimax_results*)

Format SARIMAX results as HTML.

Args:
sarimax_results (dict): The SARIMAX results.

Returns:
str: The formatted results in HTML.

`get_selected_countries`(*list_widget*)

Get the list of selected countries from a QListWidget.

Args:
list_widget (QListWidget): The list widget containing the countries.

Returns:
list: The list of selected countries.

group_countries()

Group selected countries into a single group.

load_file()

Load a CSV file and update the UI with the loaded data.

plot_selected_data()

Plot the selected data.

run_adf_test()

Run the Augmented Dickey-Fuller (ADF) test on the selected data.

run_arima(*p_range=None*, *d_range=None*, *q_range=None*)

Run the ARIMA model optimization and forecasting.

Args:

p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values.

run_arimax(*p_range=None*, *d_range=None*, *q_range=None*)

Run the ARIMAX model optimization and forecasting.

Args:

p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values.

run_sarimax(*p_range=None*, *d_range=None*, *q_range=None*, *seasonal_period=None*)

Run the SARIMAX model optimization and forecasting.

Args:

p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values. seasonal_period (int): Seasonal period.

save_forecast()

Save the forecasted data to a CSV file.

toggleSidePanel()

Toggle the visibility of the side panel(Hide the Side Panel).

update_combos()

Update combo boxes and lists with the loaded data.

update_forecasted_countries_list()

Update the list of forecasted countries.

1.6 Plotting module

Plotting.**plot_data**(*df*, *forecast_results*, *forecast_key*, *variable*, *plot_type*, *ax*, *add_forecast_start_line*, *show_confidence_interval*)

Plot data and forecasts on a matplotlib axis.

Args:

df (pd.DataFrame): Data frame containing the data. forecast_results (dict): Forecast results. forecast_key (str): Key for the forecast result. variable (str): Variable to plot. plot_type (str): Type of plot ("Historical", "Forecast", "Both"). ax (matplotlib.axes.Axes): Matplotlib axis to plot on. add_forecast_start_line (bool): Whether to add a line indicating the start of the forecast. show_confidence_interval (bool): Whether to show confidence intervals.

Returns:
float: Maximum value in the plotted data.

Plotting.**plot_data_stacked_bar**(*df*, *forecast_results*, *forecast_keys*, *variable*, *plot_type*, *ax*)

Plot stacked bar chart for data and forecasts on a matplotlib axis.

Args:
df (pd.DataFrame): Data frame containing the data. forecast_results (dict): Forecast results. forecast_keys (list): List of forecast keys. variable (str): Variable to plot. plot_type (str): Type of plot ("Historical", "Forecast", "Both"). ax (matplotlib.axes.Axes): Matplotlib axis to plot on.

Returns:
float: Maximum value in the plotted data.

1.7 Sarimax module

Sarimax.**forecast_future**(*sarimax_results*, *df*, *start_year*, *forecast_until_year=2100*, *replace_negative_forecast=False*)

Forecast future values using SARIMAX models.

Args:
sarimax_results (dict): SARIMAX results. df (pd.DataFrame): Data frame containing the data. start_year (int): Start year for the forecast. forecast_until_year (int): Year until which to forecast. replace_negative_forecast (bool): Whether to replace negative forecast values with zero.

Returns:
dict: Forecast results for each country.

Sarimax.**optimize_sarimax**(*series*, *p_range*, *d_range*, *q_range*, *seasonal_period*)

Optimize SARIMAX model parameters.

Args:
series (pd.Series): Time series data. p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values. seasonal_period (int): Seasonal period.

Returns:
tuple: Best AIC, best order, best seasonal order, best model.

Sarimax.**optimize_sarimax_models**(*adf_results*, *df*, *selected_countries*, *p_range*, *d_range*, *q_range*, *seasonal_period*, *start_year*, *end_year*)

Optimize SARIMAX models for multiple countries.

Args:
adf_results (pd.DataFrame): ADF test results. df (pd.DataFrame): Data frame containing the data. selected_countries (list): List of selected countries. p_range (range): Range of p values. d_range (range): Range of d values. q_range (range): Range of q values. seasonal_period (int): Seasonal period. start_year (int): Start year for the data. end_year (int): End year for the data.

Returns:
dict: SARIMAX results for each country.

1.8 SidePanel module

class SidePanel.**SidePanelWindow**(*main_window*)

Bases: QWidget

Side panel window for settings.

add_line()

Add a new line to the plot.

apply_arima()

Apply ARIMA model settings.

apply_arimax()

Apply ARIMAX model settings.

apply_model()

Apply the selected model settings.

apply_plot_settings()

Apply the plot settings.

apply_sarimax()

Apply SARIMAX model settings.

get_range(*text*, *default*)

Get a range of values from a text input.

Args:

text (str): The text input. default (list): The default range.

Returns:

list: The range of values.

init_correction_settings_ui()

Initialize UI elements for correction settings.

init_line_settings_ui()

Initialize UI elements for line settings.

init_model_settings_ui()

Initialize UI elements for model settings.

init_plot_settings_ui()

Initialize UI elements for plot settings.

show_model_settings()

Show the model settings UI elements.

show_plot_settings()

Show the plot settings UI elements.

update_line_list()

Update the list of active lines.

update_model_parameters(*model_name*)

Update model parameters based on selected model.

Args:

model_name (str): The name of the model.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX