**STATSMODELS**

statsmodels is a Python library that provides classes and functions for estimating and testing various statistical models. It is particularly useful for statistical modeling and hypothesis testing in the field of econometrics. Here are some key functionalities of statsmodels:

**Linear Regression**: Estimate linear regression models and perform hypothesis tests on the coefficients.

**Python : Copy code**

*import statsmodels.api as sm*

*import pandas as pd*

*# Create a DataFrame with example data*

*data = {'X': [1, 2, 3, 4, 5], 'Y': [2, 4, 5, 4, 5]}*

*df = pd.DataFrame(data)*

*# Add a constant to the independent variable (X)*

*X = sm.add\_constant(df['X'])*

*# Fit the linear regression model*

*model = sm.OLS(df['Y'], X).fit()*

*# Print the summary*

*print(model.summary())*

**Time Series Analysis:** Explore and analyze time series data, including autoregressive integrated moving average (ARIMA) models.

**Python : Copy code**

*import statsmodels.api as sm*

*import pandas as pd*

*# Create a DataFrame with time series data*

*# Example: df = pd.read\_csv('your\_time\_series\_data.csv')*

*# Fit an ARIMA model*

*model = sm.tsa.ARIMA(df['value'], order=(1, 1, 1)).fit()*

*# Print the summary*

*print(model.summary())*

Hypothesis Testing: Conduct hypothesis tests, including t-tests and F-tests.

Python : Copy code

*import statsmodels.api as sm*

*import pandas as pd*

*# Create a DataFrame with example data*

*data = {'group\_1': [25, 30, 35, 40, 45],*

*'group\_2': [20, 28, 32, 38, 42]}*

*df = pd.DataFrame(data)*

*# Perform a t-test*

*t\_stat, p\_value, \_ = sm.stats.ttest\_ind(df['group\_1'], df['group\_2'])*

*print(f'T-statistic: {t\_stat}, p-value: {p\_value}')*

Generalized Linear Models (GLM): Estimate GLMs for various distributions, such as binomial or Poisson.

Python : Copy code

*import statsmodels.api as sm*

*import pandas as pd*

*# Create a DataFrame with example data*

*data = {'X': [1, 2, 3, 4, 5],*

*'Y': [0, 1, 0, 1, 0]}*

*df = pd.DataFrame(data)*

*# Fit a logistic regression model (binomial distribution)*

*model = sm.GLM(df['Y'], sm.add\_constant(df['X']), family=sm.families.Binomial()).fit()*

*# Print the summary*

*print(model.summary())*

These are just a few examples of what statsmodels can do. It provides a comprehensive set of tools for statistical modeling and analysis, making it valuable in fields such as economics, finance, social sciences, and more.

**With prediction**

statsmodels is a versatile library that can be used for both hypothesis testing and prediction. It provides tools for estimating and testing various statistical models, including linear regression models, time series models, generalized linear models (GLMs), and more. While hypothesis testing is a significant aspect of its functionality, it also supports prediction tasks.

Here are some examples of prediction tasks using statsmodels:

Linear Regression for Prediction:

python

Copy code

import statsmodels.api as sm

import pandas as pd

# Create a DataFrame with example data

data = {'X': [1, 2, 3, 4, 5], 'Y': [2, 4, 5, 4, 5]}

df = pd.DataFrame(data)

# Add a constant to the independent variable (X)

X = sm.add\_constant(df['X'])

# Fit the linear regression model

model = sm.OLS(df['Y'], X).fit()

# Make predictions

predictions = model.predict(X)

print("Predictions:", predictions)

Time Series Forecasting:

python

Copy code

import statsmodels.api as sm

import pandas as pd

# Create a DataFrame with time series data

# Example: df = pd.read\_csv('your\_time\_series\_data.csv')

# Fit an ARIMA model

model = sm.tsa.ARIMA(df['value'], order=(1, 1, 1)).fit()

# Make predictions

forecast = model.forecast(steps=10) # Adjust the number of steps as needed

print("Forecast:", forecast)

Generalized Linear Models (GLM) for Binary Classification (Logistic Regression):

python

Copy code

import statsmodels.api as sm

import pandas as pd

# Create a DataFrame with example data

data = {'X': [1, 2, 3, 4, 5],

'Y': [0, 1, 0, 1, 0]}

df = pd.DataFrame(data)

# Fit a logistic regression model (binomial distribution)

model = sm.GLM(df['Y'], sm.add\_constant(df['X']), family=sm.families.Binomial()).fit()

# Make predictions

predictions = model.predict(sm.add\_constant(df['X']))

print("Predictions:", predictions)

In these examples, the models are fitted to the training data, and predictions are made on new or existing data. While statsmodels is powerful for hypothesis testing and model interpretation, it is important to note that for large-scale prediction tasks and machine learning applications, other libraries such as scikit-learn or TensorFlow might be more suitable.

**The ARIMA model is denoted as ARIMA(p, d, q), where:**

* �*p* is the order of the autoregressive component.
* �*d* is the degree of differencing.
* �*q* is the order of the moving average component.

**InPOPUP CREATE SCRIPT**

*import tkinter as tk*

*from tkinter import messagebox*

*# Create the main window*

*root = tk.Tk()*

*root.withdraw() # Hide the main window*

*# Create a popup*

*messagebox.showinfo("Popup Title", "Hello, this is a popup!")*

*# Start the Tkinter main loop*

*root.mainloop()*

*Seasonal arima*

*import statsmodels.api as sm*

*import pandas as pd*

*# Assume you have a pandas DataFrame 'data' with a datetime index*

*# Example: data = pd.read\_csv('your\_data.csv', index\_col='date', parse\_dates=True)*

*# Fit a Seasonal ARIMA model*

*order = (1, 1, 1) # Replace with appropriate order values*

*seasonal\_order = (1, 1, 1, 12) # Replace with appropriate seasonal order values*

*model = sm.tsa.statespace.SARIMAX(data, order=order, seasonal\_order=seasonal\_order)*

*results = model.fit()*

*# Make predictions*

*forecast\_steps = 12 # Replace with the number of steps you want to forecast*

*forecast = results.get\_forecast(steps=forecast\_steps)*

*predictions = forecast.predicted\_mean*

*# Visualize results*

*import matplotlib.pyplot as plt*

*plt.plot(data, label='Actual Data')*

*plt.plot(predictions, label='Predictions', color='red')*

*plt.legend()*

*plt.show()*