Baruch ML HW 5

Annie Yi, Daniel Tuzes, group 9

March 11, 2025

Predicting the Consumer Price Index

To predict the Consumer Price Index, we build a linear model containing features as below. We are excited to see which one of these will be have lower weights in the regression, and didn't want to neglect them before seeing the model in action. However, we have opionions based on which is caused by the inflation, and which one of these may have low correlation with the results, see the section discussion.

1. Past CPI Values: Historical CPI data to capture the persistence of inflation. Source: U.S. Bureau of Labor Statistics (BLS), Consumer Price Index for All Urban Consumers (CPI-U), U.S. city average, All items - CUUR0000SA0, `https://data.bls.gov/toppicks?survey=cu`

2. Output Gap (OG): The difference between actual and potential economic output. A positive output gap can lead to higher inflation. Source: IMF, `https://www.imf.org/en/Publications/WEO/weo-database/2024/October/download-entire-database` Note that data after 2023 are not measured but predicted, and data frequency is in years, not months.

3. Unemployment Rate (UR): Lower unemployment rates can lead to higher inflation based on the Phillips Curve. Source: U.S. Bureau of Labor Statistics (BLS), `https://data.bls.gov/toppicks?survey=ln`, Unemployment Rate - LNS14000000

4. Interest Rates (IR): Central bank policy rates, such as the Federal Funds Rate, influence inflation through monetary policy. Source: Federal Reserve Economic Data (FRED). `https://fred.stlouisfed.org/series/FEDFUNDS`

5. Money Supply (M2SL): Measures like M2 (a broad measure of money supply) can impact inflation. Source: Federal Reserve Board. `https://fred.stlouisfed.org/series/M2SL`

6. Wage Growth (WG): Increases in wages can lead to higher consumer spending and inflation. Source: Federal Reserve Bank of Atlanta. `https://www.atlantafed.org/chcs/wage-growth-tracker`, column "Overall"

Table 1: CPI and features are sourced starting from January 2015 till January 2025. The tables shows the first few entries.

date	CPI	IR	WG	CI	OG	UR	M2	ER
Jan-20	216.687	0.11	1.6	139.84	-2.326	9.8	8478	92.3566
Feb-20	216.741	0.13	1.7	136.553	-2.326	9.8	8527.6	93.7321
Mar-20	217.631	0.16	1.9	141.525	-2.326	9.9	8523.7	93.7979
Apr-20	218.009	0.2	1.9	149.313	-2.326	9.9	8555.1	92.6661
May-20	218.178	0.2	1.6	140.402	-2.326	9.6	8609.3	92.8423

7. Commodity Prices (CI): Prices of key commodities like oil and food can directly affect inflation. Source: International Monetary Fund (IMF). `https://data.imf.org/?sk=471dddf8-d8a7-499a-81ba-5b332c01f8b9&sid=1547558078595` All commodities index

8. Exchange Rates: Changes in exchange rates can influence import prices and thus inflation. Source: Federal Reserve Board, Nominal Broad Dollar Index, H10/H10/JRXWTFB_N.B `https://www.federalreserve.gov/datadownload/Download.aspx?rel=H10&series=122e3bcb627e8e53f1bf72a1a09cfb81&lastObs=&from=01/01/2010&to=02/01/2025&filetype=csv&label=include&layout=seriescolumn`

9. Consumer Confidence Index (CCI): Measures consumer sentiment and expectations about the economy. Higher confidence can lead to increased spending and inflation. Source: OECD. This feature was added later, and only its outcome is discussed. Calculation can be found in the attached notebook. `https://www.oecd.org/en/data/indicators/consumer-confidence-index-cci.html?oecdcontrol-cf46a27224-var1=USA`

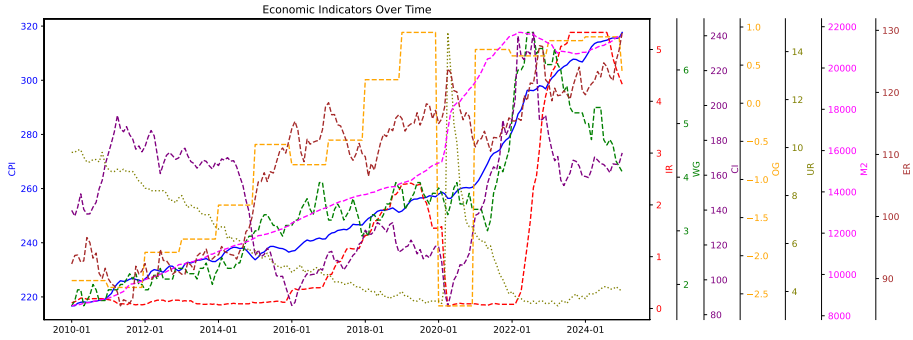


Figure 1: Economic Indicators Over Time. Note the consistent decrease in many of the indices since or around 2022, but by nature, we know that CPI can hardly decrease.

The collected data first checked for outliers and missing data points by visual inspection. Web observed no DQ issues apart from missing data. Missing data is filled by either backward or forward filling and differences in frequencies are resolved by resampling (backfilling). We can see that most of the indicators went up around 2022, and decreased since then a bit, while CPI kept increasing. This changing trend cannot be captured by the model, since this is observable only for recent years. We expect the model to not to be able to predict well after 2022.

After doing OLS and seeing how poorly the model performs, we did ridge and lasso regression at different alphas on the first 12 years of data from 2010 till 2022, and inspected the output on the whole time range, which includes data from end of 2022 till 2025. Plotted results are shown in the figures 2 and 3.

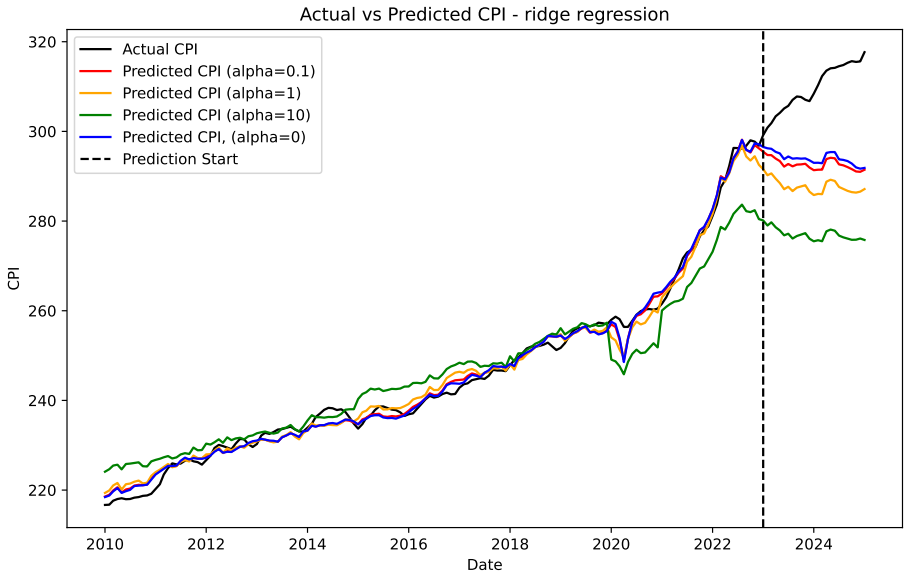


Figure 2: Predictions of the models with ridge regression. Training data ends at end of 2022.

The fitted paramters and R^2 are shown in the table 2 for both on the training set and on the test (predicted) set. We can see that none of the models were able to predict well after 2022, a simple ffill of the data would have been a better prediction.

From the ridge and lasso regression, we can see how differently they reduced the feature coefficients: lasso regression eliminates the features, while ridge regression reduces them.

The parameter α has more severe effect on MSE and R^2 in lasso regression, which is also reflected in the worsening fitting. To have a similar severity, one needs to lower the α for the lasso regression.

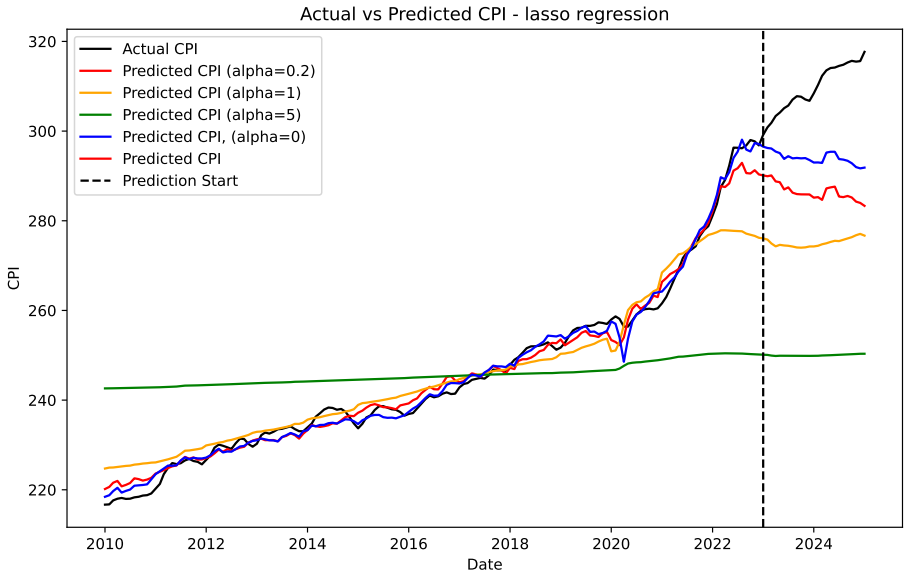


Figure 3: Predictions of the models with lasso regression. Training data ends at end of 2022.

From an econometrics point of view, if we would be satisfied with the results, we could state the following:

1. money supply (M2) and wage growth (WG) are the most important features
2. while ouput gap (OG) is not eliminated by lasso regression, it changes its sign depending on the strength and type of regularization
3. interest rate (IR), suprisingly, lost its weight in the ridge regression, and we lost it completely at moderate and strong lasso regression
4. unemployment (UR) and exchange rates (ER) had low weights, and they got eliminated by the weaker lasso regression
5. Commodity prices (CI) played a consistent, but not too strong role

It is surprising, how well the data can be fit till 2022, but breaks down after that. By extending the fitting regime till 2025, the fitted curve still wouldn't align well with the period 2022-2025.

On the other hand, by shrinking the fitted time frame till 2018, in figure 4, we can see that the model predicts the CPI well till 2022. It doesn't mean that the model would be any better, but a good example that in lucky times we can have good predictions, but this cannot be attributed to the merit of the model.

Table 2: Coefficient values and their names for ridge regression. in.cpt: axis intercept, subscript t: training set, subscript p: test set (prediction),lin reg: linear regression, rid: ridge regression, las: lasso regression

	Lin Reg	rid $\alpha_{0.1}$	rid α_1	rid α_{10}	las $\alpha_{0.2}$	las α_1	las α_5
IR	15.69	13.96	8.422	3.98	7.1	0	0
WG	14.22	13.64	16.17	13.96	19.06	1.029	0
CI	11.15	12.99	14.49	8.331	4.976	0	0
OG	-1.695	-1.128	2.143	8.041	2.403	3.091	0
UR	-7.065	-6.631	-4.207	-4.046	-0	-0	-0
M2	48.23	45.5	36.67	22.29	45.83	49.71	7.824
ER	1.909	5.746	11.91	11.5	0	0	0
in.cpt	218.2	216.8	214.5	220.9	220.9	220.9	220.9
R_t^2	0.9922	0.992	0.9874	0.9135	0.9842	0.9176	0.2161
R_p^2	-8.966	-10.43	-17.3	-38.08	-19.62	-42.03	-126.4
MSE_t	3.022	3.095	4.88	33.5	6.111	31.9	303.6
MSE_p	279.3	320.4	512.9	1095	578.1	1206	3571

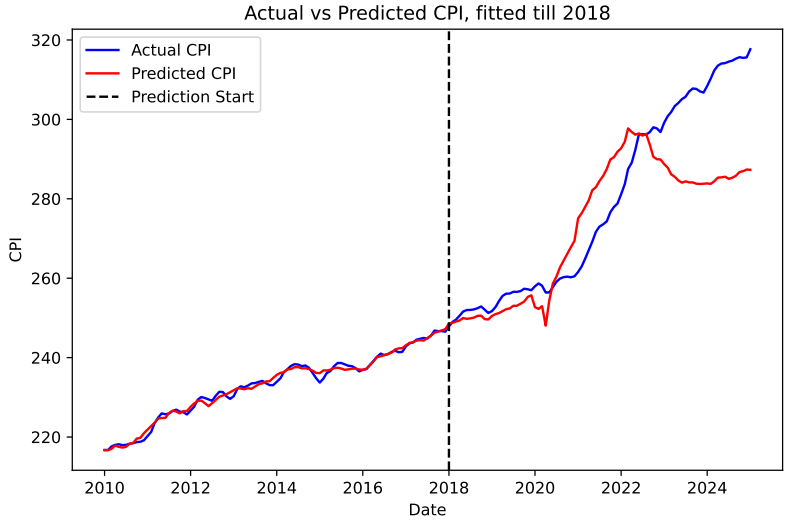


Figure 4: Predictions of the model using OLS only. Training data ends at end of 2018. The prediction looks good till 2022, but it is rather a lucky coincidence, and not a good model.

Discussion

The CPI can be well-captured by the model in a limited period of time, but the model fails to predict properly after 2022. Till 2022, only 3 features would be enough to capture most part of the curve, $R^2 = 0.97$, see attached notebook and coefficients in table 3. Here, we did lasso and ridge regression again, but starting with only 3 features.

It is instructive to see that increasing the number of paramters does not necessarily improve the quality of prediction. The model is limited by its linear nature two fold:

1. it cannot capture the time delay of the features, e.g. interest rate and money supply have a time delay effect on CPI (with different time scales),

2. it cannot capture non-linear effects, e.g. the unemployment rate, which is alredy a weak feature, and even its supposedly effect is not linear

We expected to a have a larger weight on interest rate, as it is a direct tool of the Federal Reserve to control inflation, but this regression didn't conculde that. We believe that bringing in more features can not help in giving more meaningful predictions without building a more thourough financial model, because either the changes in the new features are coused by or directly related to the inflation (like a subset of goods, different basket, etc), and they are not causing the inflation (however, the goal of such model would be to predict it), or the time delay of the features cannot be captured by such a linear model.

However, CCI, the costumer confidence index, may be a good candidate in predicting the CPI. Such parameter relies rather on a psychological factor, and the expectations of the consumers and corporates may become self-fulfilling. However, we cannot know how strong effect of the existing CPI prediction models and their advertised outcome has on the CCI.

After obtaining the data and completing the regressions, we have seen that the CPI is not a good predictor of inflation, see the table 4.

A good model relies more on a carefully selected set of features, and an appropiate modelling approach, than on the number of features.

fitting

March 11, 2025

1 CPI prediction

[3]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

1.1 Data gathering

Data was downloaded from the source noted in the text body of the document, manully copied into a file "joined.csv".

1.1.1 OG: output gap

[ ]:
```
OG = pd.read_csv("data/output_gap.csv")
OG["date"] = pd.to_datetime(OG["year"], format="%Y")
OG
```

[163]:
```
OG_monthly = OG.set_index("date").resample("MS").ffill().reset_index().
↪drop(columns="year")
OG_monthly.to_csv("data/OG.csv", index=False)
```

1.1.2 Currencies

[164]:
```
cxc = pd.read_csv("data/FRB_H10.csv", na_values=["ND"])
cxc = cxc.bfill()
cxc = cxc.rename(columns = {"JRXWTFB_N.B":"ER"})
cxc["Time Period"] = pd.to_datetime(cxc["Time Period"], format="%d/%m/%Y")
cxc = cxc.set_index("Time Period").resample("MS").ffill().reset_index()
cxc
```

[165]:
```
cxc.to_csv("data/ER.csv", index=False)
```

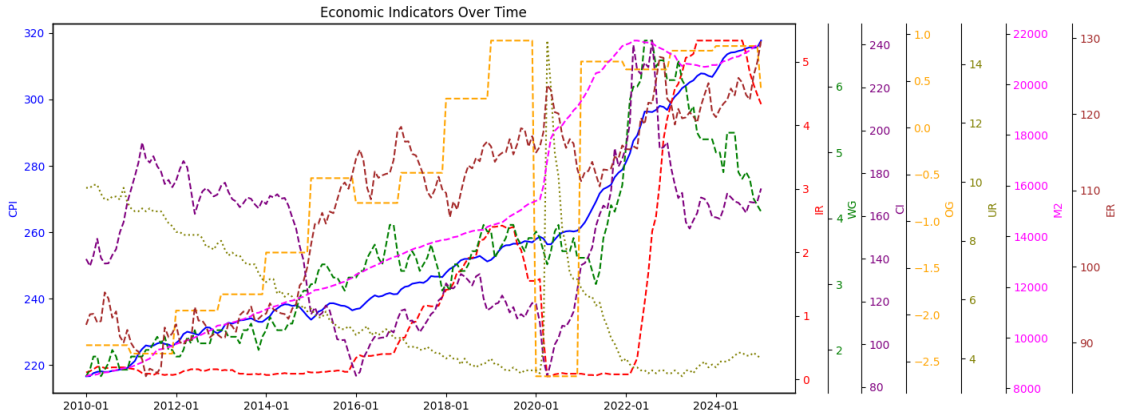
1.1.3 plot and show

[4]:
```
df = pd.read_csv("data/joined.csv")
if "CCI" in df.columns:
    df = df.drop(columns=["CCI"])
latex_table = df.head().to_latex(
    caption="CPI and features are sourced starting from January 2015 till␣
 ↪January 2025. "
            "The tables shows the first few entries.",
    index=False,
    float_format="%.6g")
print(latex_table)

# Convert the date column to datetime format
df["date"] = pd.to_datetime(df["date"], format="%b-%y")

# Set the date as the index
df.set_index("date", inplace=True)

# Plot with multiple y-axes
fig, ax1 = plt.subplots(figsize=(12, 6))

# First axis (CPI)
ax1.plot(df.index, df["CPI"], label="CPI", color="blue")
ax1.set_ylabel("CPI", color="blue")
ax1.tick_params(axis="y", labelcolor="blue")

# axis (FEDFUND IR)
ax2 = ax1.twinx()
ax2.plot(df.index, df["IR"], label="IR", color="red", linestyle="dashed")
ax2.set_ylabel("IR", color="red")
ax2.tick_params(axis="y", labelcolor="red")

# axis (Overall wage growth, WG)
ax3 = ax1.twinx()
ax3.spines["right"].set_position(("outward", 30))
ax3.plot(df.index, df["WG"], label="WG", color="green", linestyle="dashed")
ax3.set_ylabel("WG", color="green")
ax3.tick_params(axis="y", labelcolor="green")

# axis (Commodity index, CI)
ax4 = ax1.twinx()
ax4.spines["right"].set_position(("outward", 60))
ax4.plot(df.index, df["CI"], label="CI", color="purple", linestyle="dashed")
ax4.set_ylabel("CI", color="purple")
ax4.tick_params(axis="y", labelcolor="purple")
```

```
# axis (Output gap, OG)
ax4 = ax1.twinx()
ax4.spines["right"].set_position(("outward", 100))
ax4.plot(df.index, df["OG"], label="OG", color="orange", linestyle="dashed")
ax4.set_ylabel("OG", color="orange")
ax4.tick_params(axis="y", labelcolor="orange")

# axis (Unemployment rate, UR)
ax4 = ax1.twinx()
ax4.spines["right"].set_position(("outward", 150))
ax4.plot(df.index, df["UR"], label="UR", color="olive", linestyle="dotted")
ax4.set_ylabel("UR", color="olive")
ax4.tick_params(axis="y", labelcolor="olive")

# axis (Money supply, M2)
ax4 = ax1.twinx()
ax4.spines["right"].set_position(("outward", 190))
ax4.plot(df.index, df["M2"], label="M2", color="magenta", linestyle="dashed")
ax4.set_ylabel("M2", color="magenta")
ax4.tick_params(axis="y", labelcolor="magenta")

# axis (Exchange rates, ER)
ax4 = ax1.twinx()
ax4.spines["right"].set_position(("outward", 250))
ax4.plot(df.index, df["ER"], label="ER", color="brown", linestyle="dashed")
ax4.set_ylabel("ER", color="brown")
ax4.tick_params(axis="y", labelcolor="brown")


# Formatting date axis
ax1.xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.xticks(rotation=45)

plt.title("Economic Indicators Over Time")
plt.savefig("data_raw.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

```
\begin{table}
\caption{CPI and features are sourced starting from January 2015 till January
2025. The tables shows the first few entries.}
\begin{tabular}{lrrrrrrrr}
\toprule
date & CPI & IR & WG & CI & OG & UR & M2 & ER \\
\midrule
Jan-10 & 216.687 & 0.11 & 1.6 & 139.84 & -2.326 & 9.8 & 8478 & 92.3566 \\
Feb-10 & 216.741 & 0.13 & 1.7 & 136.553 & -2.326 & 9.8 & 8527.6 & 93.7321 \\
Mar-10 & 217.631 & 0.16 & 1.9 & 141.525 & -2.326 & 9.9 & 8523.7 & 93.7979 \\
Apr-10 & 218.009 & 0.2 & 1.9 & 149.313 & -2.326 & 9.9 & 8555.1 & 92.6661 \\
```

```
May-10 & 218.178 & 0.2 & 1.6 & 140.402 & -2.326 & 9.6 & 8609.3 & 92.8423 \\
\bottomrule
\end{tabular}
\end{table}
```



1.1.4 Scale and split

[8]:
```
y = df['CPI']
X = df.drop(['CPI'], axis=1)
# scale all features to be between 0 and 1
X = (X - X.min()) / (X.max() - X.min())
# drop data in X and y after 2023 jan
X_fit = X[X.index < "2023-01-01"]  # fit only here
y_fit = y[y.index < "2023-01-01"]
x_valid = X[X.index >= "2023-01-01"] # used for prediction
y_valid = y[y.index >= "2023-01-01"]
```

1.2 fitting without lasso or ridge

[9]:
```
linmodel = LinearRegression(fit_intercept=True)
linmodel.fit(X_fit, y_fit)
mse_values = {}

# Create a DataFrame for coefficients
result = pd.DataFrame(linmodel.coef_.flatten(), index=X_fit.columns,␣
 ↪columns=["Lin Reg"])

intercept_value = linmodel.intercept_
r_squared_value = linmodel.score(X_fit, y_fit)

full_pred = linmodel.predict(X)
```

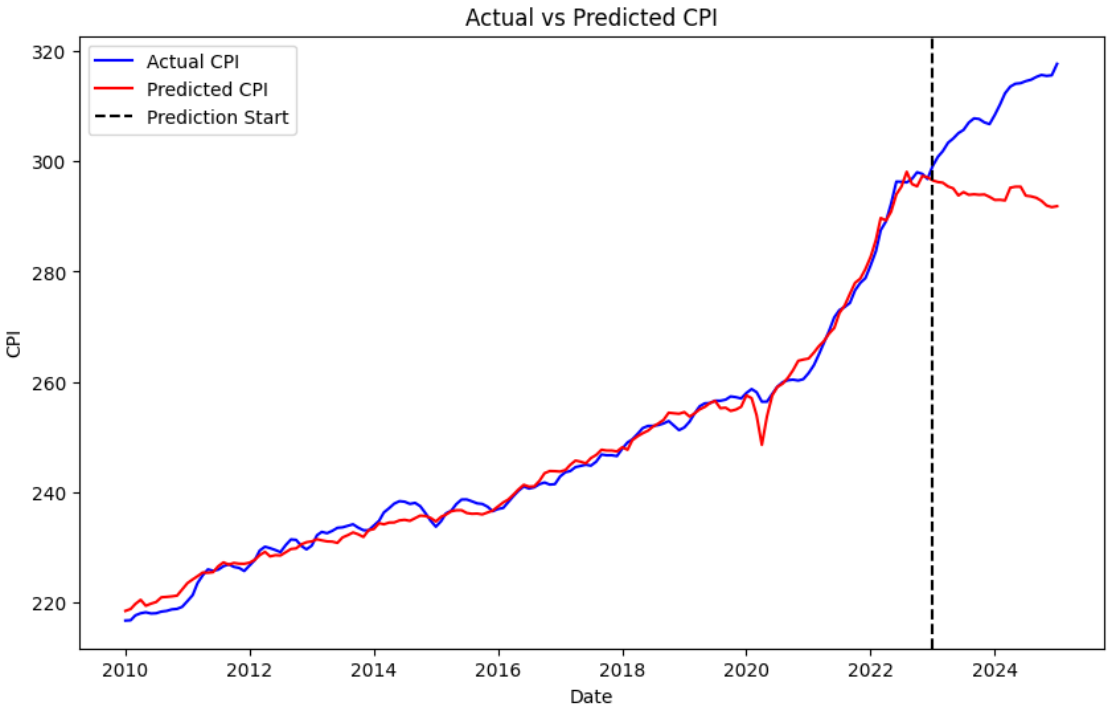
```
mse_train = mean_squared_error(y_fit, linmodel.predict(X_fit))
mse_predicted = mean_squared_error(y_valid, linmodel.predict(x_valid))
r_squared_train = linmodel.score(X_fit, y_fit)
r_squared_predicted = linmodel.score(x_valid, y_valid)

result.loc["in.cpt"] = [intercept_value]
result.loc["$R^2_t$"] = [r_squared_train]
result.loc["$R^2_p$"] = [r_squared_predicted]
result.loc["$MSE_t$"] = [mse_train]
result.loc["$MSE_p$"] = [mse_predicted]

# Print LaTeX output
latex_output = result.to_latex(caption="Coefficient values and their names")
print(latex_output)
```

```
\begin{table}
\caption{Coefficient values and their names}
\begin{tabular}{lr}
\toprule
 & Lin Reg \\
\midrule
IR & 15.693951 \\
WG & 14.219535 \\
CI & 11.149360 \\
OG & -1.694559 \\
UR & -7.065261 \\
M2 & 48.228987 \\
ER & 1.908637 \\
in.cpt & 218.195622 \\
$R^2_t$ & 0.992197 \\
$R^2_p$ & -8.965529 \\
$MSE_t$ & 3.022217 \\
$MSE_p$ & 279.333791 \\
\bottomrule
\end{tabular}
\end{table}
```

1.2.1 Show predicted vs actual

[11]:
```
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='blue')
plt.plot(pd.Series(full_pred, index=y.index), label='Predicted CPI', color='red')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
```

```
plt.title('Actual vs Predicted CPI')
plt.legend()
plt.show()
```



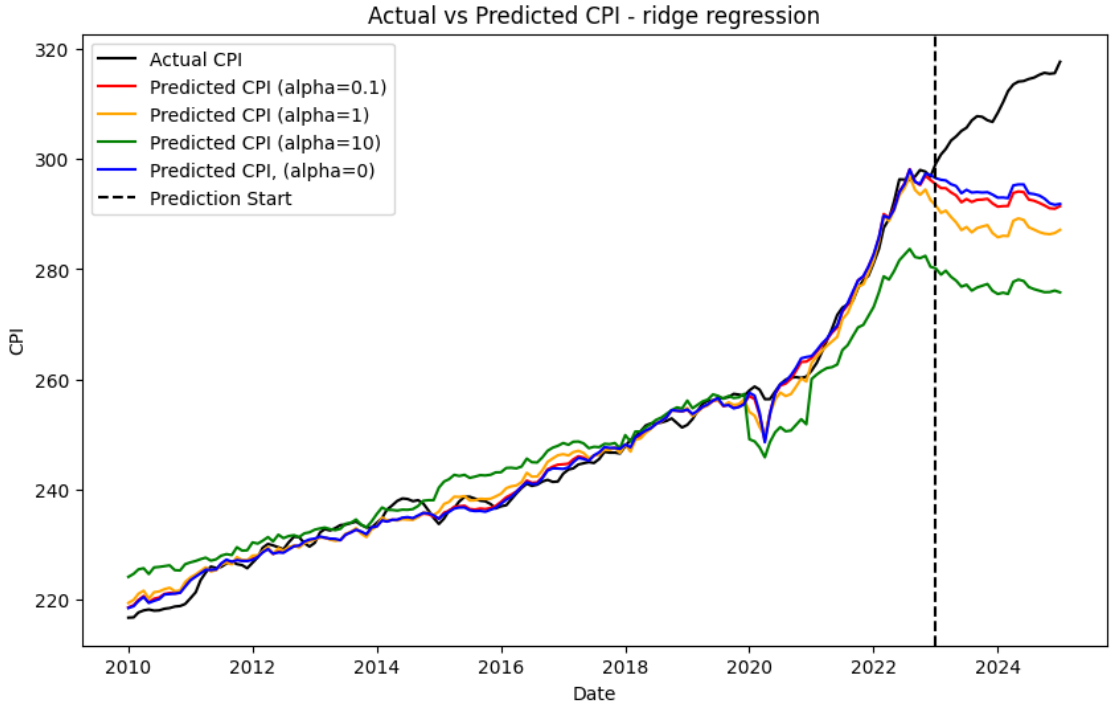
1.3 Ridge regression

[15]:
```
# do 3 ridge gression with alpha 0.1, 1 and 10
alphas = [0.1, 1, 10]
model = {}
pred = {}
ridge_result = result.copy()
for alpha in alphas:
    model[alpha] = Ridge(alpha=alpha)
    model[alpha].fit(X_fit, y_fit)
    pred[alpha] = model[alpha].predict(X)
    _result = pd.DataFrame(model[alpha].coef_.flatten(), index=X_fit.columns,␣
 ↪columns=[f"rid $\\alpha_{{{alpha}}}$"])

    intercept_value = model[alpha].intercept_

    mse_train = mean_squared_error(y_fit, model[alpha].predict(X_fit))
    mse_predicted = mean_squared_error(y_valid, model[alpha].predict(x_valid))
    r_squared_train = model[alpha].score(X_fit, y_fit)
```

```
    r_squared_predicted = model[alpha].score(x_valid, y_valid)

    _result.loc["in.cpt"] = [intercept_value]
    _result.loc["$R^2_t$"] = [r_squared_train]
    _result.loc["$R^2_p$"] = [r_squared_predicted]
    _result.loc["$MSE_t$"] = [mse_train]
    _result.loc["$MSE_p$"] = [mse_predicted]

    print("feature coefficients for alpha", alpha, ":", model[alpha].coef_)

    ridge_result = ridge_result.merge(_result, left_index=True, right_index=True)
print(ridge_result.to_latex(caption="Coefficient values and their names for␣
 ↪ridge regression"))

# and plot all on 1 graph
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='black')
colors = ['red', 'orange', 'green']
for alpha, color in zip(alphas, colors):
    plt.plot(pd.Series(pred[alpha], index=y.index), label=f'Predicted CPI␣
 ↪(alpha={alpha})', color=color)
plt.plot(pd.Series(linmodel.predict(X), index=y.index), label='Predicted CPI,␣
 ↪(alpha=0)', color='blue')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI - ridge regression')
plt.legend()
plt.savefig("data_pred_ridge.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

```
feature coefficients for alpha 0.1 : [13.95919526 13.64108809 12.9865159
-1.12791193 -6.63112417 45.49897131
  5.74569799]
feature coefficients for alpha 1 : [ 8.42200109 16.16735226 14.49063849
2.14261476 -4.20706146 36.66720024
 11.91430427]
feature coefficients for alpha 10 : [ 3.98023712 13.96429636  8.33119899
8.04082269 -4.04626582 22.28717481
 11.4992021 ]
\begin{table}
\caption{Coefficient values and their names for ridge regression}
\begin{tabular}{lrrrr}
\toprule
 & Lin Reg & rid $\alpha_{0.1}$ & rid $\alpha_{1}$ & rid $\alpha_{10}$ \\
\midrule
```

```
IR & 15.693951 & 13.959195 & 8.422001 & 3.980237 \\
WG & 14.219535 & 13.641088 & 16.167352 & 13.964296 \\
CI & 11.149360 & 12.986516 & 14.490638 & 8.331199 \\
OG & -1.694559 & -1.127912 & 2.142615 & 8.040823 \\
UR & -7.065261 & -6.631124 & -4.207061 & -4.046266 \\
M2 & 48.228987 & 45.498971 & 36.667200 & 22.287175 \\
ER & 1.908637 & 5.745698 & 11.914304 & 11.499202 \\
in.cpt & 218.195622 & 216.807650 & 214.542920 & 220.908152 \\
$R^2_t$ & 0.992197 & 0.992008 & 0.987399 & 0.913501 \\
$R^2_p$ & -8.965529 & -10.431181 & -17.298698 & -38.080626 \\
$MSE_t$ & 3.022217 & 3.095251 & 4.880338 & 33.502171 \\
$MSE_p$ & 279.333791 & 320.416003 & 512.912498 & 1095.429948 \\
\bottomrule
\end{tabular}
\end{table}
```

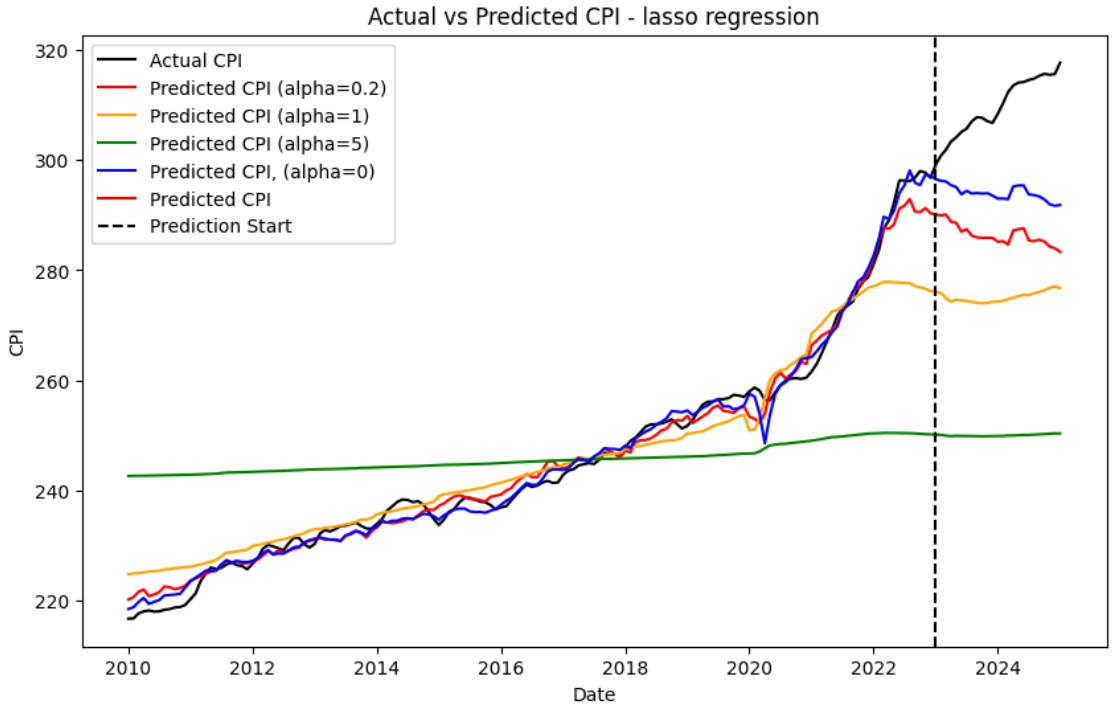


1.3.1 lasso regression

[17]:
```
# do 3 lasso gression with alpha 0.1, 1 and 10
alphas = [0.2, 1, 5]
model = {}
pred = {}
lasso_ridge_result = ridge_result.copy()
```

```
for alpha in alphas:
    model[alpha] = Lasso(alpha=alpha)
    model[alpha].fit(X_fit, y_fit)
    pred[alpha] = model[alpha].predict(X)
    _result = pd.DataFrame(model[alpha].coef_.flatten(), index=X_fit.columns,␣
 ↪columns=[f"las $\\alpha_{{{alpha}}}$"])

    mse_train = mean_squared_error(y_fit, model[alpha].predict(X_fit))
    mse_predicted = mean_squared_error(y_valid, model[alpha].predict(x_valid))
    r_squared_train = model[alpha].score(X_fit, y_fit)
    r_squared_predicted = model[alpha].score(x_valid, y_valid)

    _result.loc["in.cpt"] = [intercept_value]
    _result.loc["$R^2_t$"] = [r_squared_train]
    _result.loc["$R^2_p$"] = [r_squared_predicted]
    _result.loc["$MSE_t$"] = [mse_train]
    _result.loc["$MSE_p$"] = [mse_predicted]

    lasso_ridge_result = lasso_ridge_result.merge(_result, left_index=True,␣
 ↪right_index=True)
print(lasso_ridge_result.to_latex(
    caption=("Coefficient values and their names for ridge regression. "
             "in.cpt: axis intercept, subscript t: training set, "
             "subscript p: test set (prediction),lin reg: linear regression, "
             "rid: ridge regression, las: lasso regression"),
    label="tab:ridge_coefficients",
    float_format="%.4g"))

# and plot all on 1 graph
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='black')
colors = ['red', 'orange', 'green']
for alpha, color in zip(alphas, colors):
    plt.plot(pd.Series(pred[alpha], index=y.index), label=f'Predicted CPI␣
 ↪(alpha={alpha})', color=color)
plt.plot(pd.Series(linmodel.predict(X), index=y.index), label='Predicted CPI,␣
 ↪(alpha=0)', color='blue')
plt.plot(pd.Series(pred, index=y.index), label='Predicted CPI', color='red')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI - lasso regression')
plt.legend()
plt.savefig("data_pred_lasso.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

```
\begin{table}
\caption{Coefficient values and their names for ridge regression. in.cpt: axis
intercept, subscript t: training set, subscript p: test set (prediction),lin
reg: linear regression, rid: ridge regression, las: lasso regression}
\label{tab:ridge_coefficients}
\begin{tabular}{lrrrrrrr}
\toprule
 & Lin Reg & rid $\alpha_{0.1}$ & rid $\alpha_{1}$ & rid $\alpha_{10}$ & las
$\alpha_{0.2}$ & las $\alpha_{1}$ & las $\alpha_{5}$ \\
\midrule
IR & 15.69 & 13.96 & 8.422 & 3.98 & 7.1 & 0 & 0 \\
WG & 14.22 & 13.64 & 16.17 & 13.96 & 19.06 & 1.029 & 0 \\
CI & 11.15 & 12.99 & 14.49 & 8.331 & 4.976 & 0 & 0 \\
OG & -1.695 & -1.128 & 2.143 & 8.041 & 2.403 & 3.091 & 0 \\
UR & -7.065 & -6.631 & -4.207 & -4.046 & -0 & -0 & -0 \\
M2 & 48.23 & 45.5 & 36.67 & 22.29 & 45.83 & 49.71 & 7.824 \\
ER & 1.909 & 5.746 & 11.91 & 11.5 & 0 & 0 & 0 \\
in.cpt & 218.2 & 216.8 & 214.5 & 220.9 & 220.9 & 220.9 & 220.9 \\
$R^2_t$ & 0.9922 & 0.992 & 0.9874 & 0.9135 & 0.9842 & 0.9176 & 0.2161 \\
$R^2_p$ & -8.966 & -10.43 & -17.3 & -38.08 & -19.62 & -42.03 & -126.4 \\
$MSE_t$ & 3.022 & 3.095 & 4.88 & 33.5 & 6.111 & 31.9 & 303.6 \\
$MSE_p$ & 279.3 & 320.4 & 512.9 & 1095 & 578.1 & 1206 & 3571 \\
\bottomrule
\end{tabular}
\end{table}
```



1.4 select best model

[167]:
```
remove_columns = []
# keep_cols = ["CPI", "IR", "WG", "CI", "OG", "UR", "M2", "ER"]
keep_cols = ["CPI", "WG", "OG", "M2"]
remove_columns = [col for col in df.columns if col not in keep_cols]
# remove_columns = ["date", "CPI", "IR", "WG", "CI", "OG", "UR", "M2", "ER"]
# remove_columns = ["UR", "ER"]
# remove_columns = ["ER", "M2", "UR"]
```

[171]:
```
y = df['CPI']
X = df.drop(['CPI', *remove_columns], axis=1)
# scale all features to be between 0 and 1
X = (X - X.min()) / (X.max() - X.min())
# drop data in X and y after 2023 jan
X_fit = X[X.index < "2023-01-01"]  # fit only here
y_fit = y[y.index < "2023-01-01"]
x_pred = X[X.index >= "2023-01-01"] # used for prediction
y_pred = y[y.index >= "2023-01-01"]
X_fit
```

[172]:
```
linmodel_s = LinearRegression(fit_intercept=True)
linmodel_s.fit(X_fit, y_fit)

# Create a DataFrame for coefficients
result = pd.DataFrame(linmodel_s.coef_.flatten(), index=X_fit.columns,␣
↪columns=["Lin Reg"])

# Store intercept and R-squared as additional rows
intercept_value = linmodel_s.intercept_
r_squared_value = linmodel_s.score(X_fit, y_fit)

result.loc["Intercept"] = [intercept_value]
result.loc["R-squared"] = [r_squared_value]

# Print LaTeX output
latex_output = result.to_latex(caption="Coefficient values and their names")
print(latex_output)
```

[173]:
```
# do 3 ridge gression with alpha 0.1, 1 and 10
from sklearn.linear_model import Ridge
```

```
alphas = [0.1, 1, 10]
model = {}
pred = {}
ridge_result_s = result.copy()
for alpha in alphas:
    model[alpha] = Ridge(alpha=alpha)
    model[alpha].fit(X_fit, y_fit)
    pred[alpha] = model[alpha].predict(X)
    _result = pd.DataFrame(model[alpha].coef_.flatten(), index=X_fit.columns,␣
 ↪columns=[f"rid $\\alpha_{{{alpha}}}$"])

    intercept_value = model[alpha].intercept_
    r_squared_value = model[alpha].score(X_fit, y_fit)

    _result.loc["Intercept"] = [intercept_value]
    _result.loc["R-squared"] = [r_squared_value]

    print("feature coefficients for alpha", alpha, ":", model[alpha].coef_)

    ridge_result_s = ridge_result_s.merge(_result, left_index=True,␣
 ↪right_index=True)
print(ridge_result_s.to_latex(caption="Coefficient values and their names for␣
 ↪ridge regression"))

# and plot all on 1 graph
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='black')
colors = ['red', 'orange', 'green']
for alpha, color in zip(alphas, colors):
    plt.plot(pd.Series(pred[alpha], index=y.index), label=f'Predicted CPI␣
 ↪(alpha={alpha})', color=color)
plt.plot(pd.Series(linmodel_s.predict(X), index=y.index), label='Predicted CPI,␣
 ↪(alpha=0)', color='blue')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI - ridge regression')
plt.legend()
plt.savefig("data_pred_ridge_s.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

[174]:
```
# do 3 lasso gression with alpha 0.1, 1 and 10
from sklearn.linear_model import Lasso
alphas = [0.2, 1, 5]
model = {}
pred = {}
lasso_ridge_result_s = ridge_result_s.copy()
for alpha in alphas:
    model[alpha] = Lasso(alpha=alpha)
    model[alpha].fit(X_fit, y_fit)
    pred[alpha] = model[alpha].predict(X)
    _result = pd.DataFrame(model[alpha].coef_.flatten(), index=X_fit.columns,␣
 ↪columns=[f"las $\\alpha_{{{alpha}}}$"])

    intercept_value = model[alpha].intercept_
    r_squared_value = model[alpha].score(X_fit, y_fit)

    _result.loc["Intercept"] = [intercept_value]
    _result.loc["R-squared"] = [r_squared_value]

    # print("feature coefficients for alpha", alpha, ":", model[alpha].coef_)

    lasso_ridge_result_s = lasso_ridge_result_s.merge(_result, left_index=True,␣
 ↪right_index=True)
print(lasso_ridge_result_s.to_latex(caption="Coefficient values and their names␣
 ↪for ridge regression",
    label="tab:ridge_coefficients",
    float_format="%.4g"))

# and plot all on 1 graph
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='black')
colors = ['red', 'orange', 'green']
for alpha, color in zip(alphas, colors):
    plt.plot(pd.Series(pred[alpha], index=y.index), label=f'Predicted CPI␣
 ↪(alpha={alpha})', color=color)
plt.plot(pd.Series(linmodel_s.predict(X), index=y.index), label='Predicted CPI,␣
 ↪(alpha=0)', color='blue')
plt.plot(pd.Series(pred, index=y.index), label='Predicted CPI', color='red')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI - lasso regression')
plt.legend()
plt.savefig("data_pred_lasso_s.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

1.5 add CCI

[197]:
```
df = pd.read_csv("data/joined.csv")
latex_table = df.head().to_latex(
    caption="CPI and features are sourced starting from January 2015 till␣
 ↪January 2025. "
            "The tables shows the first few entries.",
    index=False,
    float_format="%.6g")
print(latex_table)

# Convert the date column to datetime format
df["date"] = pd.to_datetime(df["date"], format="%b-%y")

# Set the date as the index
df.set_index("date", inplace=True)
```

[198]:
```
y = df['CPI']
X = df.drop(['CPI'], axis=1)
# scale all features to be between 0 and 1
X = (X - X.min()) / (X.max() - X.min())
# drop data in X and y after 2023 jan
X_fit = X[X.index < "2023-01-01"]  # fit only here
y_fit = y[y.index < "2023-01-01"]
x_pred = X[X.index >= "2023-01-01"] # used for prediction
y_pred = y[y.index >= "2023-01-01"]
```

[207]:
```
linmodel = LinearRegression(fit_intercept=True)
linmodel.fit(X_fit, y_fit)
mse_values = {}

# Create a DataFrame for coefficients
result = pd.DataFrame(linmodel.coef_.flatten(), index=X_fit.columns,␣
 ↪columns=["Lin Reg"])

# Store intercept and R-squared as additional rows
intercept_value = linmodel.intercept_
r_squared_value = linmodel.score(X_fit, y_fit)

pred = linmodel.predict(X)
pred_mse = linmodel.predict(X_fit)
mse = mean_squared_error(y_fit, pred_mse)
```

```
result.loc["in.cpt"] = [intercept_value]
result.loc["$R^2$"] = [r_squared_value]
result.loc["MSE"] = [mse]

# Print LaTeX output
latex_output = result.to_latex(caption="Coefficient values and their names")
print(latex_output)
```

[208]:
```
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='blue')
plt.plot(pd.Series(pred, index=y.index), label='Predicted CPI', color='red')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI')
plt.legend()
plt.show()
```

[212]:
```
# do 3 ridge gression with alpha 0.1, 1 and 10
from sklearn.linear_model import Ridge
alphas = [0.1, 1, 10]
model = {}
pred = {}
ridge_result = result.copy()
for alpha in alphas:
    model[alpha] = Ridge(alpha=alpha)
    model[alpha].fit(X_fit, y_fit)
    pred[alpha] = model[alpha].predict(X)
    _result = pd.DataFrame(model[alpha].coef_.flatten(), index=X_fit.columns,␣
↪columns=[f"rid $\\alpha_{{{alpha}}}$"])

    intercept_value = model[alpha].intercept_
    r_squared_value = model[alpha].score(X_fit, y_fit)

    pred_mse = linmodel.predict(X_fit)
    mse = mean_squared_error(y_fit, pred_mse)
    result.loc["MSE"] = [mse]
    _result.loc["in.cpt"] = [intercept_value]
    _result.loc["$R^2$"] = [r_squared_value]

    print("feature coefficients for alpha", alpha, ":", model[alpha].coef_)
```

```
    ridge_result = ridge_result.merge(_result, left_index=True, right_index=True)
print(ridge_result.to_latex(caption="Coefficient values and their names for␣
 ↪ridge regression"))

# and plot all on 1 graph
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='black')
colors = ['red', 'orange', 'green']
for alpha, color in zip(alphas, colors):
    plt.plot(pd.Series(pred[alpha], index=y.index), label=f'Predicted CPI␣
 ↪(alpha={alpha})', color=color)
plt.plot(pd.Series(linmodel.predict(X), index=y.index), label='Predicted CPI,␣
 ↪(alpha=0)', color='blue')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI - ridge regression')
plt.legend()
plt.savefig("data_pred_ridge_s.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

[215]:
```
# do 3 lasso gression with alpha 0.1, 1 and 10
from sklearn.linear_model import Lasso
alphas = [0.2, 1, 5]
model = {}
pred = {}
lasso_ridge_result = ridge_result.copy()
for alpha in alphas:
    model[alpha] = Lasso(alpha=alpha)
    model[alpha].fit(X_fit, y_fit)
    pred[alpha] = model[alpha].predict(X)
    _result = pd.DataFrame(model[alpha].coef_.flatten(), index=X_fit.columns,␣
 ↪columns=[f"las $\\alpha_{{{alpha}}}$"])

    intercept_value = model[alpha].intercept_
    r_squared_value = model[alpha].score(X_fit, y_fit)

    pred_mse = linmodel.predict(X_fit)
    mse = mean_squared_error(y_fit, pred_mse)
    result.loc["MSE"] = [mse]
    _result.loc["in.cpt"] = [intercept_value]
    _result.loc["$R^2$"] = [r_squared_value]
```

```
    lasso_ridge_result = lasso_ridge_result.merge(_result, left_index=True,␣
 ↪right_index=True)
lasso_ridge_result.drop(index=["IR", "WG", "CI", "OG", "UR", "M2", "ER"],␣
 ↪inplace=True)
print(lasso_ridge_result.to_latex(caption="Coefficient values and their names␣
 ↪for ridge regression",
    label="tab:ridge_coefficients",
    float_format="%.4g"))

# and plot all on 1 graph
plt.figure(figsize=(10, 6))
plt.plot(y, label='Actual CPI', color='black')
colors = ['red', 'orange', 'green']
for alpha, color in zip(alphas, colors):
    plt.plot(pd.Series(pred[alpha], index=y.index), label=f'Predicted CPI␣
 ↪(alpha={alpha})', color=color)
plt.plot(pd.Series(linmodel.predict(X), index=y.index), label='Predicted CPI,␣
 ↪(alpha=0)', color='blue')
plt.plot(pd.Series(pred, index=y.index), label='Predicted CPI', color='red')
plt.axvline(pd.to_datetime("2023-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI - lasso regression')
plt.legend()
plt.savefig("data_pred_lasso_s.pdf", format="pdf", bbox_inches="tight")
plt.show()
```

1.6 fitting till 2018

[4]:
```
y = df['CPI']
X = df.drop(['CPI'], axis=1)
# scale all features to be between 0 and 1
X = (X - X.min()) / (X.max() - X.min())
# drop data in X and y after 2023 jan
X_fit = X[X.index < "2018-01-01"]  # fit only here
y_fit = y[y.index < "2018-01-01"]
x_pred = X[X.index >= "2018-01-01"] # used for prediction
y_pred = y[y.index >= "2018-01-01"]
```

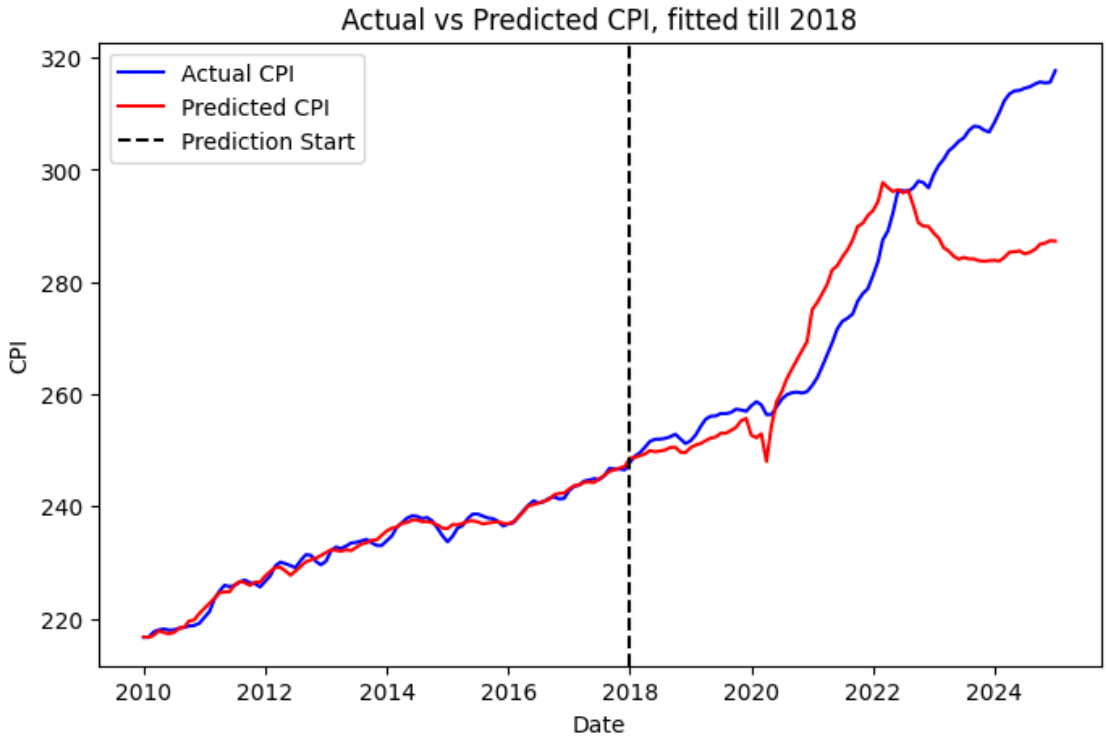
[5]:
```
linmodel = LinearRegression(fit_intercept=True)
linmodel.fit(X_fit, y_fit)
mse_values = {}
```

```
# Create a DataFrame for coefficients
result = pd.DataFrame(linmodel.coef_.flatten(), index=X_fit.columns,␣
 ↪columns=["Lin Reg"])

intercept_value = linmodel.intercept_
r_squared_value = linmodel.score(X_fit, y_fit)

pred = linmodel.predict(X)
pred_mse = linmodel.predict(X_fit)
mse = mean_squared_error(y_fit, pred_mse)

result.loc["in.cpt"] = [intercept_value]
result.loc["$R^2$"] = [r_squared_value]
result.loc["MSE"] = [mse]

# Print LaTeX output
latex_output = result.to_latex(caption="Coefficient values and their names")
print(latex_output)
```

```
\begin{table}
\caption{Coefficient values and their names}
\begin{tabular}{lr}
\toprule
 & Lin Reg \\
\midrule
IR & -2.515663 \\
WG & 2.515347 \\
CI & 11.921843 \\
OG & 3.516296 \\
UR & -9.850090 \\
M2 & 65.058352 \\
ER & -3.688137 \\
in.cpt & 218.390430 \\
$R^2$ & 0.989074 \\
MSE & 0.740401 \\
\bottomrule
\end{tabular}
\end{table}
```

[9]:
```
plt.figure(figsize=(8, 5))
plt.plot(y, label='Actual CPI', color='blue')
plt.plot(pd.Series(pred, index=y.index), label='Predicted CPI', color='red')
plt.axvline(pd.to_datetime("2018-01-01"), color='black', linestyle='--',␣
 ↪label='Prediction Start')
plt.xlabel('Date')
```

```
plt.ylabel('CPI')
plt.title('Actual vs Predicted CPI, fitted till 2018')
plt.legend()
plt.savefig("data_pred_till_2018.pdf")
plt.show()
```



[ ]:

Table 3: Coefficient values and their names for ridge regression. The model is shrunk to 3 features. in.cpt: axis intercept, lin reg: linear regression, rid: ridge regression, las: lasso regression

	Lin Reg	rid $\alpha_{0.1}$	rid α_1	rid α_{10}	las $\alpha_{0.2}$	las α_1	las α_5
WG	30.38	29.99	27.37	17.75	24.52	1.029	0
OG	3.245	3.53	5.64	11.26	3.215	3.091	0
M2	42.02	41.77	39.42	25.87	43.55	49.71	7.824
in.cpt	218.4	218.5	219.2	224.6	219.6	224.5	242.6
R^2	0.974	0.9739	0.9711	0.8826	0.9717	0.9176	0.2161
MSE	10.08	10.1	11.18	45.46	10.95	31.9	303.6

Table 4: Coefficient values and their names for ridge regression

	Lin Reg	rid $\alpha_{0.1}$	rid α_1	rid α_{10}	las $\alpha_{0.2}$	las α_1	las α_5
CCI	4.729	3.165	-3.095	-6.137	-0	-0	-0
in.cpt	212.5	213.1	217.8	225.7	218.1	224.5	242.6
R^2	0.9927	0.9925	0.9864	0.9216	0.9842	0.9176	0.2161