Cal Grant Predictor

Authors: Vladimir Kataev, Darshan Patel, Harjas Dhaliwal

CS105- Winter 2023

Introduction

As college students, we recognize the importance and urgency of financial aid programs that have a direct impact on individuals and families who are facing difficulties in paying for the increasing expenses of education. Many students belong to low to middle-income backgrounds which present significant economic challenges when pursuing higher education. For many students, financial aid programs are the difference between attending college and being forced to forego higher education due to financial constraints.

Our dataset came from the California Student Aid Commission (CSAC), It is comprised of several different datasets including FAFSA and Cal Grant applications, as well as data on financial aid awards and student demographics such as expected family contribution (EFC), income level, and educational background. The dataset has many different attributes, like the ones listed above however a large portion of them were hidden observations, such as GPA, Parental Income, and Age. This dataset includes numbers from several years ago (since the start of the Cal Grant Program in 2000) before we began to clean the data. Thus, making it easier to build a predictive model because we have enough data to follow trends and build predictions. Making it a valuable resource for analyzing trends and patterns in financial aid distribution. We chose to focus on the past two years of the dataset to maintain relevance and accuracy in our analysis.

https://www.csac.ca.gov/data-dashboards

Offered grants / Total applications

Install all the frameworks

Here we Installed all of the necessary framework and toolkits to effectively implement and utilize predictive analytics techniques in our time series forecasting model. We then gathered and preprocessed the relevant data, including historical financial aid application numbers and demographic and economic data.

```
!pip install skforecast
!pip install statsmodels
from google.colab import files
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

# Modeling and Forecasting
# ==============================================================================
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.tree import DecisionTreeRegressor

from skforecast.ForecasterAutoreg import ForecasterAutoreg
from skforecast.ForecasterAutoregCustom import ForecasterAutoregCustom
from skforecast.ForecasterAutoregDirect import ForecasterAutoregDirect
from skforecast.model_selection import grid_search_forecaster
from skforecast.model_selection import backtesting_forecaster
from skforecast.utils import save_forecaster
from skforecast.utils import load_forecaster
```

```
from statsmodels.tsa.arima.model import ARIMA
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting skforecast
  Downloading skforecast-0.7.0-py2.py3-none-any.whl (343 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 343.8/343.8 KB 6.4 MB/s eta 0:00:00
Collecting optuna<3.2,>=2.10.0
  Downloading optuna-3.1.0-py3-none-any.whl (365 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 365.3/365.3 KB 14.2 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn<1.3,>=1.0 in /usr/local/lib/python3.9/dist-packages (from skforecast) (1.2.2)
Requirement already satisfied: joblib<1.3.0,>=1.1.0 in /usr/local/lib/python3.9/dist-packages (from skforecast) (1.1.1)
Requirement already satisfied: numpy<1.25,>=1.20 in /usr/local/lib/python3.9/dist-packages (from skforecast) (1.22.4)
Collecting tqdm<4.65,>=4.57.0
  Downloading tqdm-4.64.1-py2.py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.5/78.5 KB 4.8 MB/s eta 0:00:00
Requirement already satisfied: pandas<1.6,>=1.2 in /usr/local/lib/python3.9/dist-packages (from skforecast) (1.4.4)
Collecting colorlog
  Downloading colorlog-6.7.0-py2.py3-none-any.whl (11 kB)
Collecting alembic>=1.5.0
  Downloading alembic-1.10.2-py3-none-any.whl (212 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.2/212.2 KB 12.7 MB/s eta 0:00:00
Requirement already satisfied: PyYAML in /usr/local/lib/python3.9/dist-packages (from optuna<3.2,>=2.10.0->skforecast) (6.0)
Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.9/dist-packages (from optuna<3.2,>=2.10.0->skfore
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from optuna<3.2,>=2.10.0->skforeca
Collecting cmaes>=0.9.1
  Downloading cmaes-0.9.1-py3-none-any.whl (21 kB)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas<1.6,>=1.2->skforecast) (2
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas<1.6,>=1.2->skfo
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn<1.3,>=1.0->
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn<1.3,>=1.0->skforeca
Collecting Mako
  Downloading Mako-1.2.4-py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.7/78.7 KB 3.6 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/python3.9/dist-packages (from alembic>=1.5.0->optuna<3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas<1.6,>
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.9/dist-packages (from sqlalchemy>=1.3.0->optuna<3.
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.9/dist-packages (from Mako->alembic>=1.5.0->optun
Installing collected packages: tqdm, Mako, colorlog, cmaes, alembic, optuna, skforecast
  Attempting uninstall: tqdm
    Found existing installation: tqdm 4.65.0
    Uninstalling tqdm-4.65.0:
      Successfully uninstalled tqdm-4.65.0
Successfully installed Mako-1.2.4 alembic-1.10.2 cmaes-0.9.1 colorlog-6.7.0 optuna-3.1.0 skforecast-0.7.0 tqdm-4.64.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: statsmodels in /usr/local/lib/python3.9/dist-packages (0.13.5)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.9/dist-packages (from statsmodels) (1.4.4)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.9/dist-packages (from statsmodels) (23.0)
Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.9/dist-packages (from statsmodels) (1.10.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from statsmodels) (1.22.4)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.9/dist-packages (from statsmodels) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.25->statsmod
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.25->statsmodels) (2022
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
```

The marginal probabilities file

After consolidating our datafile and creating sums 3D coordinates for the number of Cal grant A,B,C. We performed a series of data analyses to identify patterns and trends in the demand for financial aid programs.

```
print("The marginal file")
files.upload()
df = pd.read_csv('/content/marginals.csv')
# place = {CCC, CSU, UC}, parentalEducation = {"college" (or more), "highSchool" (or less), }
```

```
The marginal file
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving marginals.csv to marginals.csv
```

```
df.head()
```

	place	parentalEducation	studentYear	cga	cgb	cgc	applications
0	CCC	college	freshman	8419	53272	402	150532
1	CCC	college	sophomore	7565	32227	680	77343
2	CCC	college	junior	8020	4017	226	20014

Now we find the probability of a grant

Here we implemented an algorithm for calculating the probabilities of each grant being awarded based on various factors such as parental education level, college type applying to, and year of entrance into college. This algorithm can help identify which grants are most likely to be awarded to a particular student, based on their unique circumstances and characteristics.

```
possibleParental = ["college", "highSchool", "unknown"]
possiblePlace = ["CCC", "UC", "CSU"]
year = ["freshman", 'sophomore', 'junior', 'senior']


numeratorA  = 0
numeratorB  = 0
numeratorC  = 0
denominator = 0

print("Parental education (1-College or more, 2-High School or less, 3-unknown, 4-refuse to answer)")
p = input()
print("Place of education (1-CCC, 2-UC, 3-CSU, 4-refuse to answer)")
e = input()
print("Year of education (1-freshman, 2-sophomore, 3-junior, 4-senior, 5-refuse to answer)")
y = input()

if(p == "1"):
  possibleParental = ["college"]
elif(p == "2"):
  possibleParental = ["highSchool"]
elif(p == "3"):
  possibleParental = ["unknown"]
else:
  possibleParental = ["college", "highSchool", "unknown"]

if(e == "1"):
  possiblePlace = ["CCC"]
elif(e == "2"):
  possiblePlace = ["UC"]
elif( e == "3"):
  possiblePlace = ["CSU"]
else:
  possiblePlace = ["CCC", "UC", "CSU"]


if(y == "1"):
  year = ["freshman"]
elif(y == "2"):
  year = ["sophomore"]
elif(y == "3"):
  year = ["junior"]
elif(y == "4"):
  year = ["senior"]
else:
  year = ["freshman", 'sophomore', 'junior', 'senior']



print(possibleParental, possiblePlace, year)
```

```
Parental education (1-College or more, 2-High School or less, 3-unknown, 4-refuse to answer)
1
Place of education (1-CCC, 2-UC, 3-CSU, 4-refuse to answer)
3
Year of education (1-freshman, 2-sophomore, 3-junior, 4-senior, 5-refuse to answer)
2
['college'] ['CSU'] ['sophomore']
```

This algorithm is designed to calculate the probabilities associated with receiving various forms of financial aid.

```
import math
for yr in year:
  for pl in possiblePlace:
    for par in possibleParental:
      rel = df[df["place"] == pl]
      rel = rel[rel[" studentYear"] == yr]
      rel = rel[rel[" parentalEducation"] == par]
      numeratorA += int(rel[" cga"])
      numeratorB += int(rel[" cgb"])
      numeratorC += int(rel[" cgc"])
      denominator += int(rel[" applications"])


n1 = (numeratorA / denominator)
n2 = (numeratorB / denominator)
n3 = (numeratorC / denominator)
print("P(grant A):", float(n1) , " P(grant B):", float(n2) , " P(grant C):", float(n3))
```

```
P(grant A): 0.18002622562497356  P(grant B): 0.2724504039592234  P(grant C): 0.0
```

The time series prediction

```
print("The time series file")
files.upload()
```

```
The time series file
```

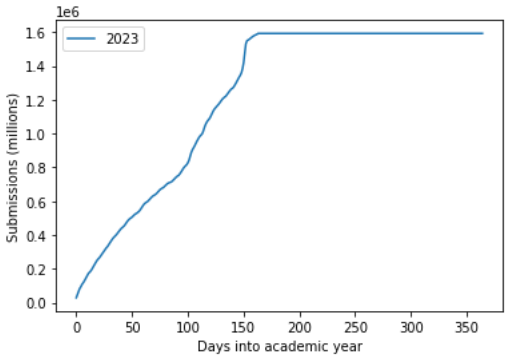
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Total_Apps_Vert_copy.csv to Total_Apps_Vert_copy.csv
{'Total_Apps_Vert_copy.csv': b'Month of Date App,Day of Date
App,2021,2022,2023\r\nOctober,1,56661,19470,28969\r\nOctober,2,80084,40330,47464\r\nOctober,3,96039,56940,65801\r\nOctober,4
```

The visualization below depicts current number of 2023 applications for financial aid.

```
data = pd.read_csv("Total_Apps_Vert_copy.csv")
data['index'] = data.index
data["applied"] = data["2023"].diff()


fig, ax = plt.subplots()
ax.plot(data['2023'], label = "2023")
plt.ylabel("Submissions (millions)")
plt.xlabel("Days into academic year")
ax.legend()
```

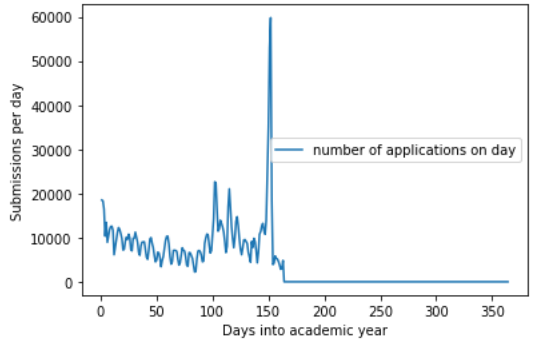
```
<matplotlib.legend.Legend at 0x7efd68ab8940>
```



The graph below visually displays the number of applications received each day, up until the present time. The point where the line on the graph levels off indicates precisely when we would initiate our Arima model to make predictions about future application rates

```
fig, ax = plt.subplots()

ax.plot(data['applied'], label = "number of applications on day")
plt.ylabel("Submissions per day")
plt.xlabel("Days into academic year")

ax.legend()
```

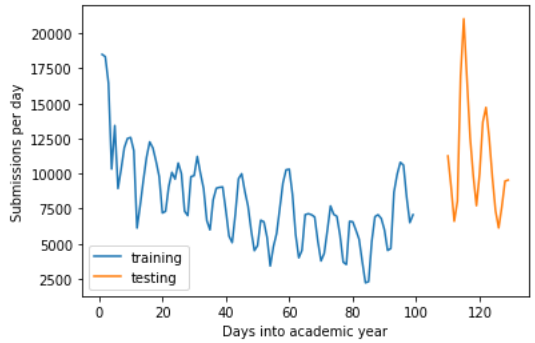
```
<matplotlib.legend.Legend at 0x7efd6612f550>
```



The visualization below demonstrates the technique allows for the comparison of testing and training data sets to evaluate the performance of our prediction algorithm. The gap in the trendline was implemented so that the testing was done on the future and not the next day.

```
train = data[0:100]
test = data[110:130]
train.head()

fig, ax = plt.subplots()
train['applied'].plot(ax=ax, label = "training")
test['applied'].plot(ax=ax, label = "testing")
plt.ylabel("Submissions per day")
plt.xlabel("Days into academic year")

ax.legend()
```

```
<matplotlib.legend.Legend at 0x7efda0345400>
```



This program implements Arima model predicts how many applications will be filled out per day for the remainder of 2023. By utilizing historical data and applying predictive analytics techniques, we can accurately forecast the demand for financial aid programs.

```
ARIMAmodel = ARIMA(train['applied'], order=(10,0,0))
ARIMAmodel = ARIMAmodel.fit()
```

```
/usr/local/lib/python3.9/dist-packages/statsmodels/base/model.py:604: ConvergenceWarning: Maximum Likelihood optimization fa
  warnings.warn("Maximum Likelihood optimization failed to "
```

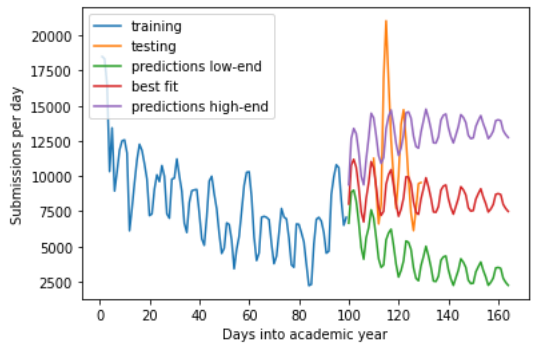
```
steps = 165 - 100
predictions = ARIMAmodel.get_forecast(steps=steps)
predictions = predictions.conf_int(alpha = 0.15)
predictions.head()
```

	lower applied	upper applied
100	6661.585555	9388.866691
101	8866.875700	12670.567473
102	9001.875216	13389.157211
103	8224.354549	13020.864984

Here, we present the formal expansion of the prediction sum for students who are expected to apply through the end of this year. The training set is displayed until April, and subsequent predictions cover from April onwards. We emphasize that the best-fit line represents optimal accuracy, while both high and low ends denote possible boundaries in terms of student application rates.

```
fig, ax = plt.subplots()
train['applied'].plot(ax=ax, label = "training")
test['applied'].plot(ax=ax, label = "testing")
#predictions[predictions["lower applied"] < 0] = 0
predictions["lower applied"].plot(ax=ax, label = 'predictions low-end')

predictions["mle"] = (predictions["upper applied"] + predictions["lower applied"]) / 2
predictions["mle"].plot(ax = ax, label = "best fit")
predictions["upper applied"].plot(ax=ax, label = 'predictions high-end')
plt.ylabel("Submissions per day")
plt.xlabel("Days into academic year")

ax.legend()
```

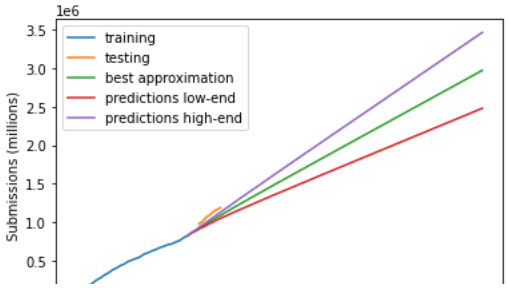
```
<matplotlib.legend.Legend at 0x7efd63f03580>
```



We see that the high-end is somewhat accurat, so from now we use that for our predictions for the whole year

```
fig, ax = plt.subplots()


predictions = ARIMAmodel.get_forecast(365 - 100)
predictions = predictions.conf_int(alpha = 0.6)


train['2023'].plot(ax=ax, label = "training")
test['2023'].plot(ax=ax, label = "testing")
predictions = predictions.cumsum()
predictions["lower applied"] += train["2023"][99]
predictions["upper applied"] += train["2023"][99]

predictions["mle"] = (predictions["upper applied"] + predictions["lower applied"]) / 2



plt.ylabel("Submissions (millions)")
plt.xlabel("Days into academic year")

predictions["mle"].plot(ax=ax, label = "best approximation")
predictions["lower applied"].plot(ax=ax, label = 'predictions low-end')
predictions["upper applied"].plot(ax=ax, label = 'predictions high-end')
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7efd65fe3e50>
```



In this instance, we are conducting an estimation on the overall quantity of submissions for financial aid that will be finalized by September 30th 2023, which is just about 2.94 million submissions.

```
predictions.tail()
#predictions.cumsum()
```

	lower applied	upper applied	mle
360	2.454842e+06	3.425798e+06	2.940320e+06
361	2.460869e+06	3.435712e+06	2.948291e+06
362	2.466871e+06	3.445603e+06	2.956237e+06
363	2.472905e+06	3.455524e+06	2.964214e+06
364	2.478902e+06	3.465409e+06	2.972155e+06

In this section, we endeavored to utilize K-Nearest Neighbors (KNN) algorithm in a univariate setting. Specifically, we partitioned the age variable into bins and computed its corresponding values for applicants versus those who were awarded admission. Consequently, we developed a prediction model that estimates the likelihood of obtaining acceptance based on one's age at application or enrollment status as a freshman, sophomore, junior or senior year student.

```
points = [[17, 122365 / 375653], [19, 66257/533635], [22, 77589 / 664747], [27, 44042 / 326286], [30, 53021 / 516651 ]] #dream ac

print("enter your age, with decimal point")
age = input()
age = float(age)


nearestPt = points[1]

for point in points:
  if abs(point[0] - age) < abs(nearestPt[0] - age):
    nearestPt = point

print("Your 1-nearest point, age-wise, has p(offered a grant) = ", float(nearestPt[1]))
```

```
enter your age, with decimal point
23.7
Your 1-nearest point, age-wise, has p(offered a grant) =  0.11671959407112781
```

```
points = [[1, 245063 / 689576], [2, 166303/261075], [3, 120873/250014], [4,65213/ 158463]]

print("enter your year, with decimal point")
year = input()
year = float(year)


nearestPt = points[1]

for point in points:
  if abs(point[0] - year) < abs(nearestPt[0] - year):
    nearestPt = point

print("Your 1-nearest point, year-wise, has p(offered a grant) = ", float(nearestPt[1]))
```

```
enter your year, with decimal point
3.5
Your 1-nearest point, year-wise, has p(offered a grant) =  0.48346492596414603
```

Conclusion

Based on our time series forecasting using Arima, we have determined that the total number of FAFSA+CADAA applications for this year is expected to fall within a range of (2478902,3465409). This forecast indicates an increase in application numbers compared to last year's count of 2416972 and the preceding year's count of 2527237. It can be concluded that there will likely be more applications submitted than in previous years. If you are looking to receive a Cal Grant A, it is recommended to enroll as a freshman at one of the University of California campuses. This recommendation is since UC campuses have historically been awarded more Cal Grant A's than California State University or private institutions. This information can help applicants make more informed decisions about where to apply and potentially increase their chances of receiving financial aid. We hope this analysis provides useful insights and guidance for those seeking financial aid in California.