Introdução

Utilizamos uma instância EC2 da AWS para executar três scripts feitos na aula de Inteligência Artificial e Aprendizado de Máquina. Durante a execução, verificamos o consumo de dois recursos principalmente, CPU e RAM. As conclusões foram registradas no estudo.

Scripts utilizados

iaam_250318_projeto_k_fold.py

Código feito em aula com o intuito de mostrar a técnica de validação cruzada (k-fold), utilizado uma base de dados de uma concessionária. Nesta base temos dados do preço, km rodados, idade do modelo e se ele foi vendido ou não.

Utilizamos um modelo de árvore de decisão e um dummy classifier para servir de base de comparação para a precisão do modelo. Depois disso, o Cross Validation foi utilizado para validar de forma mais assertiva.

iaam_250325_predição ouber.py

Utilizamos uma base de dados fantasia da Uber, feito pelo professor para termos ideas de análise que poderiam ser feitas no projeto do semestre.

Durante a fase de tratamento de dados, foram utilizadas ferramentas como o HotLabelEncoder para tornar colunas como categorias de corridas em números que podem ser utilizados.

Além disso, o seaborn foi utilizado para gerar gráficos e auxiliar a visualizar o comportamento dos dados, a presença de outliers e outras análise visuais.

O modelo utilizado foi o Gradient Booster Regressor, e utilizamos as métricas: erro médio ao quadrado, erro absoluto ao quadrado, r2 score e accuracy score para verificar a precisão do modelo.

iaam_250401_analise_de_inadimplência_bancaria.py

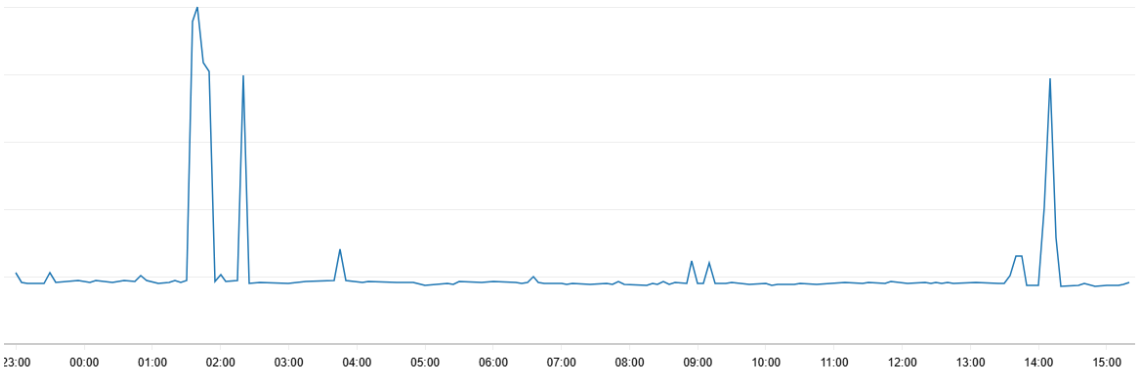
Este projeto buscou prever a chance de uma pessoa ser inadimplente em relação ao pagamento de um empréstimo. A base possuía as colunas renda_mensal, valor_financiamento, anos_trabalho, score_credito e se ele foi inadimplente ou não.

Também foram utilizadas ferramentas gráficas para visualizar os dados, além de uma matriz de correlação, para ver quais variáveis tinham relação com a outra.

O diferencial deste projeto foi a comparação entre modelos, pois utilizamos: Regressão Logística, K Nearest Neighbors, Árvore de decisão e Random Forest. Depois comparamos a precisão resultado da validação cruzada e escolhemos o mais adequado.

Resultados de monitoramento

Uso Geral da CPU:




```
ubuntu@ip-172-31-85-185:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           957Mi       369Mi       228Mi       904Ki       542Mi       587Mi
Swap:             0B          0B          0B
```

Podemos ver no gráfico gerado pela própria ferramenta da AWS, quando foram os picos de uso de CPU da instância. Os primeiros dois picos foram referentes a criação e configuração da instância, e seu uso atingiu cerca de 25% da CPU em média.

Além disso, utilizando o comando free -h, podemos ver que temos 957M de RAM total.

O primeiro pico menor após a configuração foi um teste do primeiro script (iaam_250318_projeto_k_fold.py). O uso foi de cerca de 7,14% da CPU em média. Os outros dois algoritmos foram executados em tempos próximos, e a ferramenta de análise de CPU contou como se fosse um, mas a análise detalhada de cada processo está apresentada a seguir:

1º: iaam_250318_projeto_k_fold.py

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24311	**ubuntu**	**20**	**0**	**194152**	**140392**	**55680**	**R**	**90.0**	**14.3**	**0:02.71**	**python3**
1	root	20	0	22504	13952	9600	S	0.0	1.4	0:06.66	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub_
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
15	root	20	0	0	0	0	S	0.0	0.0	0:00.52	ksoftirqd/0
16	root	20	0	0	0	0	I	0.0	0.0	0:00.79	rcu_sched
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.33	migration/0
18	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-inet_
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
24	root	20	0	0	0	0	S	0.0	0.0	0:00.01	khungtaskd
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-write
28	root	20	0	0	0	0	S	0.0	0.0	0:01.78	kcompactd0
29	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
30	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kinte
32	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kbloc
33	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-blkcg
34	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	irq/9-acpi
35	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xen-balloon
36	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-tpm_d
37	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-ata_s
38	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-md
39	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-md_bi
40	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-edac-

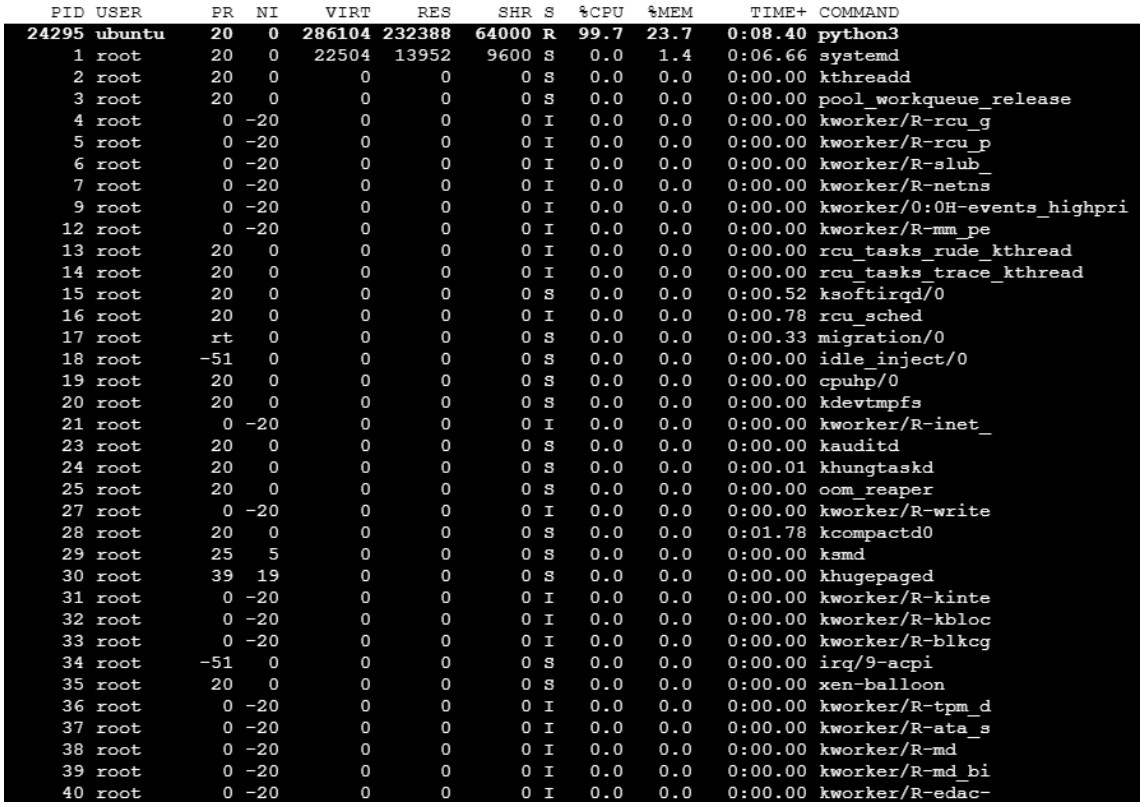
O processo mais acima, com PID 24311 é o script de k-folde rodando. Podemos perceber que o uso da CPU está focado nele, já que não tem outro processo prioritário rodando ao mesmo tempo. O uso da RAM ficou cerca de 14.3% do total

2º iaam_250325_predição ouber.py

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24288	**ubuntu**	**20**	**0**	**221768**	**169572**	**59136**	**R**	**99.9**	**17.3**	**0:04.07**	**python3**
691	_chrony	20	0	19400	3740	3072	S	0.3	0.4	0:00.98	chronyd
24277	**ubuntu**	**20**	**0**	**12372**	**5760**	**3584**	**R**	**0.3**	**0.6**	**0:00.10**	**top**
1	root	20	0	22504	13952	9600	S	0.0	1.4	0:06.64	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub_
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
15	root	20	0	0	0	0	S	0.0	0.0	0:00.52	ksoftirqd/0
16	root	20	0	0	0	0	I	0.0	0.0	0:00.78	rcu_sched
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.32	migration/0
18	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-inet_
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
24	root	20	0	0	0	0	S	0.0	0.0	0:00.01	khungtaskd
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
28	root	20	0	0	0	0	S	0.0	0.0	0:01.77	kcompactd0
29	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
30	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kinte
32	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kbloc
33	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-blkcg
34	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	irq/9-acpi
35	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xen-balloon
36	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-tpm_d
37	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-ata_s
38	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-md
39	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-md_bi

Neste script, podemos ver um uso um uso um pouco mais elevado de RAM, pois as ferramentas gráficas utilizadas consomem bastante processamento. A CPU foi praticamente priorizada apenas para a execução do código

3º iaam_250401_analise_de_inadimplência_bancaria.py



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24295	ubuntu	20	0	286104	232388	64000	R	99.7	23.7	0:08.40	python3
1	root	20	0	22504	13952	9600	S	0.0	1.4	0:06.66	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub_
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
15	root	20	0	0	0	0	S	0.0	0.0	0:00.52	ksoftirqd/0
16	root	20	0	0	0	0	I	0.0	0.0	0:00.78	rcu_sched
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.33	migration/0
18	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-inet_
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
24	root	20	0	0	0	0	S	0.0	0.0	0:00.01	khungtaskd
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-write
28	root	20	0	0	0	0	S	0.0	0.0	0:01.78	kcompactd0
29	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
30	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kinte
32	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kbloc
33	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-blkcg
34	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	irq/9-acpi
35	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xen-balloon
36	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-tpm_d
37	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-ata_s
38	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-md
39	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-md_bi
40	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-edac-

Este foi o script que mais utilizou memória RAM, o motivo disso foi o treinamento de 4 modelos diferentes, além do uso de validação cruzada, que faz várias iterações para testar o modelo.