Analisando e Predizendo a Demanda Diária em Restaurantes Universitários

Carla C. Cusihuallpa
212066
c212066@dac.unicamp.br

Diogo Bueno Moreira
228212
diogobuenomoreira@hotmail.com

Lahis G. de Almeida
228213
ra228213@g.unicamp.br

Raphael P. Santana
176414
raphaellpontess@gmail.com

***Abstract*—Este trabalho tem com objetivo a predição da demanda diária de refeições servidas no Restaurante Universitário Central da Universidade Estadual de Campinas (UNICAMP), por meio de algoritmos de Aprendizado de Máquina. O conjunto de dados foi obtido por meio da Prefeitura da UNICAMP e do CEPAGRI. Foram testados 10 modelos diferentes, obtendo os melhores resultados com os modelos *Random Forest* e Rede Neural. Nosso melhor resultado garantiu um *R2 Score* 90.67% nos testes com Redes Neurais.**

I. Introdução

O desperdício de alimentos atinge um terço de toda a comida produzida no mundo. Todos os anos, cerca de 1,3 bilhão de toneladas de alimentos são desperdiçadas em todo o mundo. O Brasil está entre os 10 principais países que mais desperdiçam comida, descartando anualmente cerca de 41 mil toneladas de alimentos [3].

Nos restaurante universitários isso não é diferente, são típicos cenários onde o desperdício de comida ocorre, seja pela falta de consciência de quem consome suas refeições quanto a forma como são planejadas. Na Universidade Estadual de Campinas (UNICAMP), na qual foi desenvolvido este estudo, a estimativa de público que irá frequentar os restaurantes em cada refeição é realizada de forma manual, baseada na experiência dos funcionários encarregados dessa tarefa [1].

Neste contexto, o trabalho desenvolvido propõe, por meio de técnicas de Aprendizado de Máquina, predizer a demanda diária dos Restaurantes Universitários da UNICAMP, contribuindo para a eficiência do planejamento do número de refeições e consequente redução do desperdício de comida.

As seções deste trabalho estão organizadas da seguinte forma: a seção II apresenta trabalhos relacionados, a seção III apresenta a base de dados utilizada bem como o preprocessamento dos dados realizado na mesma, a seção IV aponta as soluções propostas, a seção V descreve os experimentos e resultados obtidos e, por fim, a seção VI apresenta as conclusões e trabalhos futuros.

II. Trabalhos relacionados

Um trabalho relacionado foi proposto no período de 2014 [1]. Tal trabalho utilizou métodos de redes neurais obtendo-se um erro médio quadrático de 624 pessoas no conjunto de treino e 811 pessoas para conjunto de validação. Além disso, obteve predições extremas com erro de 5000 pessoas em alguns dias. O trabalho presente faz uso de comparação entre diversos modelos, para determinar qual é o melhor para o problema em questão.

No artigo [2], é realizada a apresentação de uma rede neural com somente uma camada oculta para resolver o problema na Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP). Apesar dos resultados não tão bons que obtiveram, mostrou-se a importância de usar algum método científico em substituição da forma subjetiva atual para o problema de predição de refeições dos restaurantes universitários.

III. Preprocessamento de Dados

Esta seção tem como objetivo descrever a base de dados que foi utilizada, principais correlações e encontradas e manipulações necessárias para prepará-la para ser utilizada pelos algoritmos de *machine learning*.

A. Base de Dados

A base de dados foi disponibilizada pela Prefeitura da UNICAMP. Ela é composta por 4.029 dados, incluindo café, almoço e jantar apenas do Restaurante universitário (RU) Central a partir do ano de 2012, com exceção do café, que passou a ser servido em 2016. As suas principais *features* são: **Data, Cardápio, Restaurante, Refeições Planejadas e Refeições Servidas.**

Algumas *features* consideradas relevantes foram acrescentadas na base, aumentando a chance da obtenção de um um bom resultado, como dados climáticos, que foram fornecidos pelo CEPAGRI - Centro de Pesquisa Metereológica e Climática aplicada à Agricultura; período de férias e, se o dia que foi servido a refeição, era ou não véspera de feriado. O incremento dessas novas features deve-se principalmente a observação empírica e hipóteses levantadas pelo grupo por acreditar que existir correlação entre demanda e elas. A seguir as *features* adicionadas:

- **Férias**;
- **Véspera de Feriado**;
- **Dia da Semana**;
- **Temperatura Média do Dia**;
- **Temperatura Máxima**;
- **Temperatura Mínima**;

- **Precipitação de Chuva**;
- **Umidade Relativa Máxima**;
- **Umidade Relativa Mínima**.

Além das *features* acrescentadas, algumas outras, disponibilizadas pela Prefeitura da Unicamp, foram modificadas. Este é o caso da *feature* Cardápio, onde uma normalização foi feita para ter apenas 58 pratos, eliminando dados espalhados e com valores semânticos muito similares. A *feature* Data também foi alterada, sendo separada em Ano, Mês e Dia.

Depois de realizar a normalização e a inserção das *features*, obteve-se uma base de dados com 17 *features*, porém os dados ainda não estavam prontos, devido a base de dados possuir muitas variáveis categóricas como Cardápio e Dia da Semana, que não poderiam ser utilizadas em sua forma textual nos modelos propostos. Dessa forma, o passo seguinte foi utilizar *One-Hot-Encoding*. Ao final, foi obtida uma base de dados com 130 *features*. Devemos enfatizar neste ponto que também foi aplicado *One-Hot-Encoding* nas *features* de Ano, Mês e Dia, já que são consideradas como *features* cíclicas [4].

B. Análise da Base de Dados

Na busca de conhecer melhor o conjunto de dados e determinar qual seriam os melhores algoritmos para se aplicar, uma análise foi feita, primeiro mostrando a correlação dos dados e depois um análise de cada variável com respeito ao *target* (Refeições Servidas).

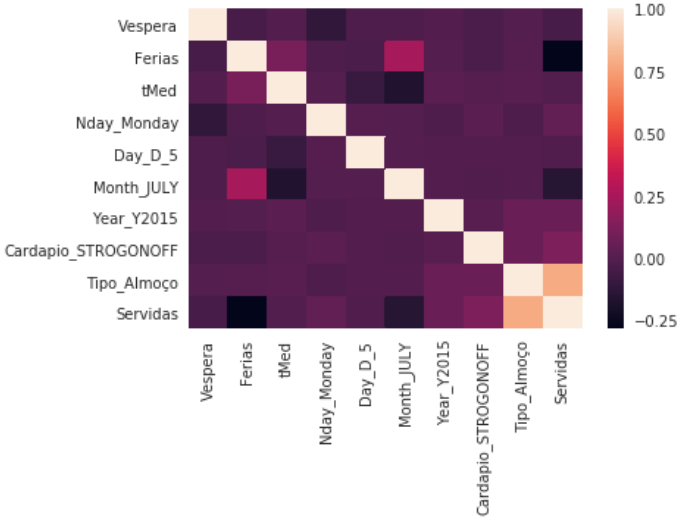

Figure 1. Correlação entre algumas *features* da base de dados.

Na Figura 1 é apresentada a correlação de algumas *features* selecionadas aleatoriamente. Pode-se observar que a *feature* Tipo_Almoço obteve a melhor correlação com a *feature target* (Servidas) enquanto que a *feature* Férias é a que tem menor correlação.

Foi feita outra análise, por meio da construção de uma árvore de decisão, com o objetivo de determinar também quais as *features* mais importantes. Na figura 2 é mostrada a árvore gerada. Da mesma forma que foi constatado na análise anterior, a *feature* Tipo_Almoço é importante para determinar o número de refeições que devem ser servidas. Entretanto, diferentemente da análise da correlação, pode-se observar que Férias também é uma *feature* relevante, assim como o *feature* Tipo_Café_da_manhã. Apesar de não se poder visualizar na figura 2, a refeição com maior aceitação pelos estudantes é o *Strogonoff*.

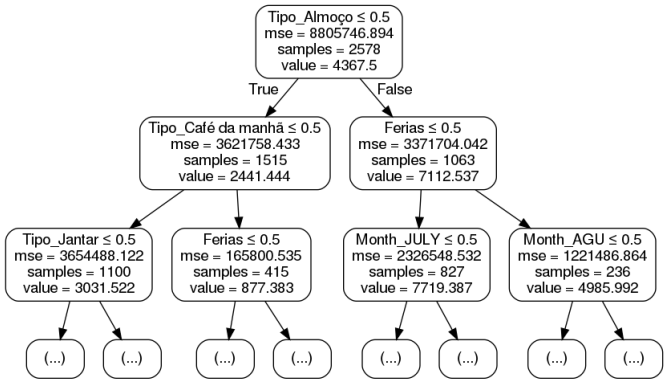

Figure 2. Árvore de decisão, mostrando os *features* mais importantes.

Portanto, das análises feitas anteriormente, pode-se perceber que para os clientes dos restaurantes é importante se vai haver Café da manhã, Almoço e Jantar; e os dias que são vésperas de feriado e as férias influenciam na demanda do restaurante, comprovando que era já esperado.

Na Figura 3, pode-se ver a distribuição dos dados entre os dias da semana. Conforme esperado, os dias que apresentam maior ocorrência para o restaurante são Terça-Feira, Quarta-Feira e Quinta-Feira, enquanto que os dias de menor ocorrência são Segunda-Feira e Sexta-Feira. Apesar disso, Segunda-Feira é o dia que possui a menor variação em quantidade de refeições, mostrando um comportamento uniforme.

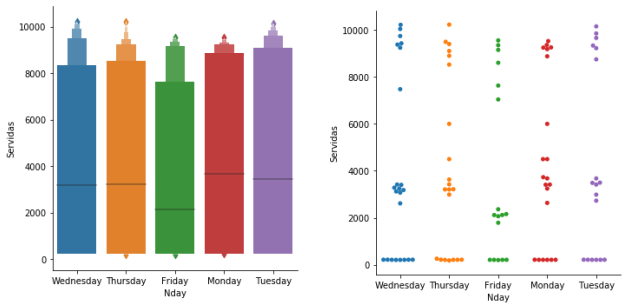

Figure 3. Distribuição de 100 dados escolhidos aleatoriamente entre os dias da semana.

Outra observação interessante é a demanda de refeição servida nas Férias e Véspera de Feriado mostradas na figura 4. Conforme esperado, nas férias a ocorrência das pessoas no restaurante é menor que em comparação com os dias normais (período de aulas), visto que a maioria

dos frequentadores no período de férias são pesquisadores e funcionários. Nos dias que são vésperas de feriado a ocorrência no restaurante também é menor que os dias comuns também, o que pode ser explicado em vista de muitos estudantes voltarem para suas casas ou simplesmente viajarem a passeio.

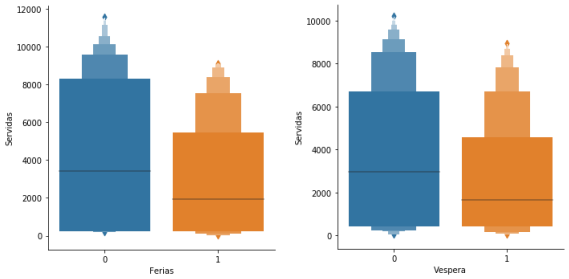

Figure 4. Distribuição de 100 dados escolhidos aleatoriamente. De cima para baixo, da direita para a esquerda: Distribuição entre o *feature* Ferias. Distribuição entre o *feature* Véspera de Feriado.

Analisando as refeições do almoço durante o ano, podemos verificar claramente sobre alguns padrões vistos anteriormente, como por exemplo, período de férias com uma demanda reduzida e em período letivo uma demanda maior. Além disso, também existe um comportamento de menor demanda em períodos de greves como pode ser visto em alguns anos.

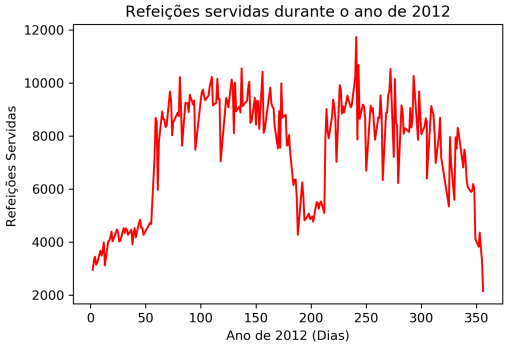

Figure 5. Gráfico anual de refeições, mostrando os períodos de maior e menor demanda.

IV. Soluções proposta

Após preprocessamento e análise dos dados, foram escolhidos dez Algoritmos para se realizar a predição da demanda diária no R.U, os quais são:

1) **Regressão Linear Simples:** o primeiro Algoritmo escolhido foi uma regressão linear simples. A Regressão Linear Simples, *Ridge* e Lasso, foram escolhidas entre todos os modelos de regressão do *sklearn* empiricamente. Todos os modelos foram testados, mas apenas estes três deram resultados satisfatórios, já que os demais demandavam alto custo computacional.

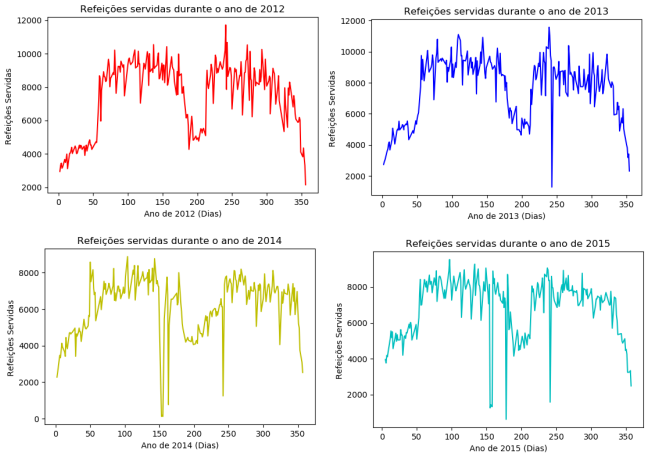

Figure 6. Gráfico anual de refeições para múltiplos anos, mostrando os períodos de maior e menor demanda.

2) **Regressão Linear *Ridge*:** O modelo faz uso da função de perda de mínimos quadrados lineares e a regularização para encontrar o melhor modelo que se ajuste ao problema. Também conhecida como regularização de *Ridge Regression* ou *Tikhonov*.
3) **Regressão Linear Lasso:** Similar a Regressão Linear *Ridge*, mas a diferença é como é feita a regularização, já que Lasso tem o mesmo mecanismo de penalização dos coeficientes com um alto grau de correlação entre si [5].
   Os três modelos descritos anteriormente são considerados como baseline deste trabalho, os demais algoritmos que serão descritos tinham como objetivo melhorar o resultado obtido pelo *baseline*.
4) **Árvore de Decisão:** É um modelo estatístico que utiliza um treinamento supervisionado para criar uma árvore para a classificação e previsão de dados, em que cada nó contêm um teste para algum atributo. Na figura 2, é possível ver um árvore de decisão para o problema da predição de demanda diária.
5) **AdaBoost com árvores de decisão e Random Forest:** O método do AdaBoost é utilizado para melhorar o desempenho de qualquer algoritmo de aprendizado, portanto, trata-se de uma técnica não utilizada de forma isolada, mas sim aplicada com outras técnicas com a finalidade de transformar classificadores fracos em fortes [9]. Assim sendo, neste trabalho combinamos o AdaBoost com o algoritmo de Árvore de Decisão e *Random Forest*.
6) **Random Forest:** trata-se de um conjunto de algoritmos de aprendizado baseados em métodos. *Random Forest* é composto por um conjunto de classificadores em forma de árvores. Cada árvore é constituída de nós e arestas. O conjunto obtido classifica novos pontos de dados através de um consenso obtido das previsões de cada classificador [6].

7) **Extreme Gradient Boost Regressor:** também conhecido como *XGBoost Regressor*, trata-se de um algoritmo recente, sendo um novo método baseado em árvore de decisão. Baseia-se na técnica do *Gradient Boosting* e visa fornecer alto desempenho sem longo tempo de computação [8].
8) **Support Vector Regression:** é uma das técnicas de aprendizado supervisionado para classificação e regressão, podendo ser linear ou não linear, ao usar-se a respectiva função do kernel [7].
9) **Rede neural de três camadas:** é uma modelo matemático que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões.

V. Experimentos e Discussão

Para implementar os modelos o linguagem de programação utilizada foi *python* v3.5.0[1]. A versão da biblioteca *sklearn* utilizada é a 19.1. Todos os modelos propostos foram executados em uma máquina com processador *Intel Core i7-3370K CPU @ 3.50Hz* × 8, com sistema operacional Ubuntu com memória de 31,3 Gb.

Para executar os modelos e observar qual foi o melhor, a base de dados foi dividida em treinamento, validação e teste, em um porcentagem de 80% para os dois primeiros e 20% para o teste. Sendo que todos os modelos foram treinados com o conjunto de treinamento (64% do total) , testando no conjunto de validação (16% do total) e nosso melhor modelo foi executado no conjunto de teste(20% do total) para obtenção do nosso resultado final. A métrica que é usada para avaliar é R2 Score.

A. Regressão Linear Simples

A biblioteca utilizada para fazer regressão linear simples foi *sklearn.linear_model.LinearRegression*. A fim de obter-se o melhor resultado, fizeram-se algumas modificações nos parâmetros, contudo, os resultados não melhoraram. Para obter os resultados parciais, utilizou-se da técnica de *K-Cross-Validation*, sendo o K escolhido igual a 10. Isso garantiu que os dados fossem diversificados, e o modelo pudesse ser analisado em diversos períodos.

Table I
Resultados finais do modelo Regressão Linear Simples.

Conjunto	Mean Square Error	R2 Score
Treinamento	912.68	90%
Validação	1392.92	76%

Os resultados finais obtidos são mostrados na tabela I, mostrando que o modelo apresentou *overfitting*, já que seu resultado reduziu muito na validação. Na figura 7 é representado algumas predições. Os pontos de cor azul são as predições realizadas pelo modelo e os pontos laranjas são os *groundtruth*. Como pode-se ver os valores do modelo estão próximos dos resultados reais, embora em sua maioria a diferença seja bem grande.

[1]https://www.python.org/

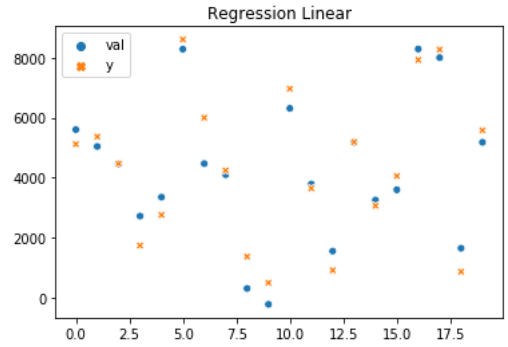

Figure 7. Alguns resultados obtidos para o modelo Regressão Linear Simples.

B. Regressão Linear Ridge

A biblioteca usada para fazer regressão linear Ridge foi *sklearn.linear_model.Ridge*. Para obter o melhor resultado, fizeram-se diversas modificações aos parâmetros como alpha e número de iterações, as quais para o melhor modelo foram 0.01 para o parâmetro alpha e 1000000 iterações. Para a obtenção desse modelo também foi utilizado o *K-Cross-Validation* com K igual a 10.

Table II
Resultados finais do modelo Regressão Linear Ridge.

Conjunto	Mean Square Error	R2 Score
Treinamento	919.70	90%
Validação	1248.79	81%

Os resultados finais obtidos para o modelo são mostrados na tabela II, em comparação com o modelo da secção V-A obteve-se melhor resultado no conjunto de validação. Na figura 8 é mostrado como o modelo fez algumas predições, similar a secção V-A. Como pode-se observar os valores do modelo estão próximos dos resultados verdadeiros e a distância é menor que as distâncias obtidas com o modelo Linear Simples. Contudo, seu erro ainda é muito grande para a maioria dos pontos preditos.

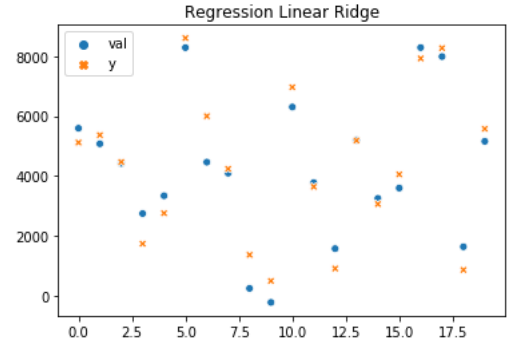

Figure 8. Alguns resultados obtidos para o modelo Regressão Linear Ridge.

C. Regressão Linear Lasso

A biblioteca usada para fazer regressão linear Lasso foi *sklearn.linear_model.Lasso*. Para obter o melhor resultado, fizeram-se as mesma modificações que para o modelo Regressão Linear *Ridge*, sendo que *alpha* foi 1 e número de iterações de 100.000. Para a obtenção desse modelo também foi utilizado o *K-Cross-Validation* com K igual a 10.

Table III
Resultados finais do modelo Regressão Linear Lasso.

Conjunto	Mean Square Error	R2 Score
Treinamento	1022.50	88%
Validação	1093.85	86%

Os resultados finais obtidos para o modelo são mostrados na tabela III. Em comparação com os modelos anteriores, o modelo avaliado não apresentou *overfitting*, mas os resultados foram piores que os modelos anteriores, além de possuir um tempo treinamento maior que os outros. Na figura 9 é mostrado como o modelo realizou algumas predições, similarmente as seções anteriores. Observa-se que os valores do modelo estão mais distante dos resultados verdadeiros frente aos anteriores.

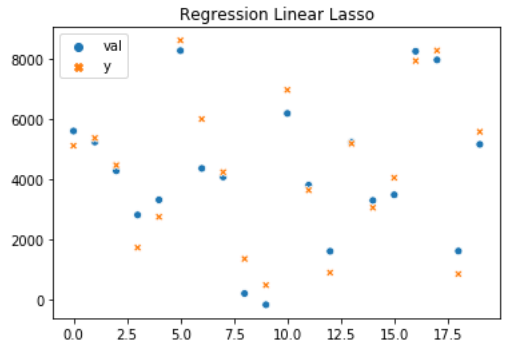

Figure 9. Alguns resultados obtidos para o modelo Regressão Linear Lasso.

Realizando uma comparação entre os três modelos obtidos, o melhor deles é o da Regressão Linear *Ridge*, sendo escolhido como *baseline* para os próximos modelos.

D. Árvore de Decisão

Nosso modelo seguinte para avaliação foi o de árvore de decisão para regressão, a implementação foi feita com a biblioteca *sklearn.tree.DecisionTreeRegressor*. Para este modelo não se fizeram modificações em seus parâmetros *defaults*.

Table IV
Resultados finais do modelo Árvores de Decisão.

Conjunto	Mean Square Error	R2 Score
Treinamento	0	100%
Validação	656.65	82.93%

Os resultados finais obtidos para o modelo são mostrados na tabela IV. A primeira observação a ser feita é a que o modelo apresentou *overfitting*. Em comparação com o *baseline*, o modelo avaliado obteve resultados piores.

E. Árvore de Decisão com AdaBoost

A implementação foi feita com as bibliotecas *sklearn.tree.DecisionTreeRegressor* para o árvore de decisão e *sklearn.ensemble.AdaBoostRegressor* para o *AdaBoost*. Fez-se modificações para *AdaBoost*, sendo o parâmetro modificado o número de estimadores que foi considerado como 300.

Table V
Resultados finais do modelo Ada Boost com Árvores de Decisão.

Conjunto	Mean Square Error	R2 Score
Treinamento	22.11	99.96%
Validação	490.72	90.88%

Os resultados finais obtidos para o modelo são mostrados na tabela V. Os resultados foram melhores do que somente a árvore de decisão, apesar de também mostrar um pouco de *overfitting*. Além disso, os resultados foram melhores que o *baseline*, possuindo um erro médio quadrático de apenas 490.72.

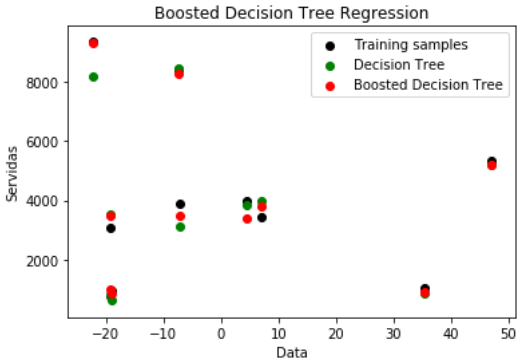

Figure 10. Alguns resultados obtidos para os modelos Árvores de Decisão e Ada Boost com Árvores de Decisão.

Na figura 10, pode-se observar algumas previsões feitas com os dois modelos estudados anteriormente. Como apresentado no gráfico, o *AdaBoost* obteve resultados que tem menor distância que os obtidos pelo modelo de Árvore de Decisão V-D, ou seja, mais próximos dos valores reais. Isso somente é garantido pela utilização do Adaboost e seus múltiplos modelos internos.

F. Random Forest

Para a implementação do algoritmo *Random Forest* para regressão, utilizou-se a biblioteca *sklearn.ensemble.RandomForestRegressor*. Para este modelo não foram feitas modificações em seus parâmetros *defaults*.

Table VI
Resultados finais do modelo Random Forest.

Conjunto	Mean Square Error	R2 Score
Treinamento	197.74	97.69%
Validação	500.87	91.42%

Os resultados finais obtidos para o modelo são mostrados na tabela VI. Em comparação com *baseline*, o modelo avaliado obteve melhores resultados e também em comparação com o modelo da seção V-E, embora a melhora tenha sido miníma.

G. Random Forest com AdaBoost

A implementação foi feita com as bibliotecas *sklearn.ensemble.RandomForestRegressor* para*Random Forest* e *sklearn.ensemble.AdaBoostRegressor* para o *AdaBoost*. Algumas modificações foram realizadas para *AdaBoost*, sendo o parâmetro modificado o número de estimadores que foi considerado como 300, similar ao da seção V-E.

Table VII
Resultados finais do modelo Random Forest com Ada Boost.

Conjunto	Mean Square Error	R2 Score
Treinamento	125.55	99.74%
Validação	502.16	91.38%

Os resultados finais obtidos para o modelo são mostrados na tabela VII. Os resultados obtidos para o modelo foram quase os mesmos que da secção anterior, sendo até mesmo um pouco piores, também pode-se observar que ocorreu *overfitting* nesse modelo. Apesar disso, apresentou melhores resultados que o *baseline* e os modelos visto nas secções V-D e V-E.

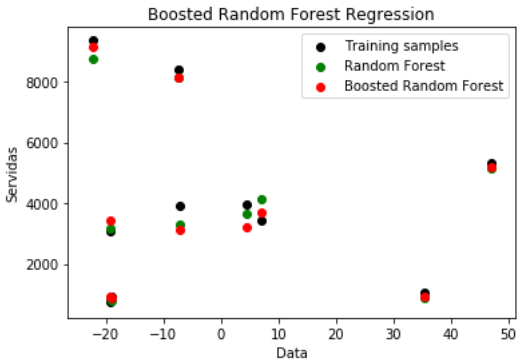

Figure 11. Alguns resultados obtidos para os modelos Random Forest e Random Forest com Ada Boost.

Na figura 11, pode-se ver algumas previsões feitas com os dois modelos estudados anteriormente. Os dois modelos obtiveram resultados semelhantes e bons.

H. Extreme Gradient Boost Regressor

Para a implementação do algoritmo *Extreme Gradient Boost Regressor* para regressão, utilizou-se a biblioteca *xgboost.XGBRegressor*[2]. Foi avaliado o modelo *Extreme Gradient Boost Regressor* conhecido como *XGBRegressor*, já que é muito usado nas competições de *Kaggle*, obtendo os melhores resultados em muitas competições.

Table VIII
Resultados finais do modelo XGBRegressor.

Conjunto	Mean Square Error	R2 Score
Treinamento	102.34	95.85%
Validação	572.52	89.88%

Os resultados finais obtidos para o modelo são mostrados na tabela VIII. Em comparação com o *baseline*, o modelo avaliado obteve melhores resultados, porém em comparação com os modelos estudados como Árvores de Decisão e *Random Forest*, não apresentou resultados melhores. Dessa forma, o algoritmo *XGBoostRegressor* para nosso problema não é a melhor técnica para ser utilizada. Na figura 12, pode-se ver algumas previsões feitas com o modelo.

Figure 12. Alguns resultados obtidos para o modelo XGBRegressor.

I. Support Vector Regression

O seguinte modelo para avaliação é o *Support Vector Machine* para regressão conhecido como *Support Vector Regression*, a implementação foi feita por meio da biblioteca *sklearn.svm.SVR*. Para esta avaliação, utilizou-se os três tipos de *kernel* que a biblioteca possui, os quais são: linear, polinómica e rgb.

Table IX
Resultados finais do modelo Super Vector Regressor.

Conjunto	Mean Square Error	R2 Score
	rbf	
Treinamento	2478.54	−3.65%
Validação	2450.71	−3.99%
	linear	
Treinamento	1902.02	35.69%
Validação	1885.37	37.18%
	polinomial	
Treinamento	2225.03	14.93%
Validação	2199.51	15.50%

[2]https://xgboost.readthedocs.io

Os resultados finais obtidos para o modelo são mostrados na tabela IX. Entre os três *kernels* avaliados que apresentou melhor resultado foi o linear. enquanto que o pior resultado foi com o *kernel* rbg. Entretanto, os resultados dos três *kernels* não superaram os obtidos com a baseline. Outro fator interessante dos resultados, foi o de que em nenhum dos *kernels* ocorreu *overfitting*. Na figura 13, pode-se ver algumas previsões feitas com o modelo para os três *kernels*. Para a maioria dos dados pode-se observar que os resultados obtidos com SVR tem uma distância muito maior em comparação com os outros modelos. Portanto, sendo o modelo que mais apresentou erro.

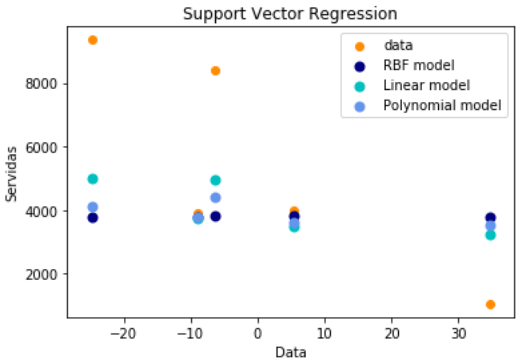

Figure 13. Alguns resultados obtidos para o modelo Suport Vector Regressor para os kernels rbg, linear e polinômico.

J. Rede neural

Para a implementação da rede neural foi utilizada a biblioteca *keras*. Para seu treinamento, foi considerado como parâmetro de perda a métrica *Mean Absolute Error*, já que a versão de *keras* que foi utilizada não possuía a métrica *R2 Score*. A estrutura da rede é de uma camada de entrada com uma função de ativação RELU com 129 neurônios, 3 camadas ocultas com função de ativação RELU com 256 neurônios em cada uma delas e uma camada de saída com função de ativação linear com um neurônio para o número predito. Apenas três camadas ocultas foram utilizadas, pois aumentando o número de camadas ocultas de 4 até 10 camadas o resultado não apresentou melhora, além do tempo de treinamento ter aumentado consideravelmente. O número de épocas estipulado foi de 200.

Table X
Resultados finais do modelo Rede Neural.

Conjunto	Mean Square Error	R2 Score
Treinamento	54.27	93.85%
Validação	482.59	91.94%

Os resultados finais obtidos para o modelo são mostrados na tabela X e na figura 14 a função de perda com respeito ao número de épocas. Em comparação com os modelo avaliados até aqui, a rede neural foi o melhor modelo obtido, apesar da melhora não ter sido muito

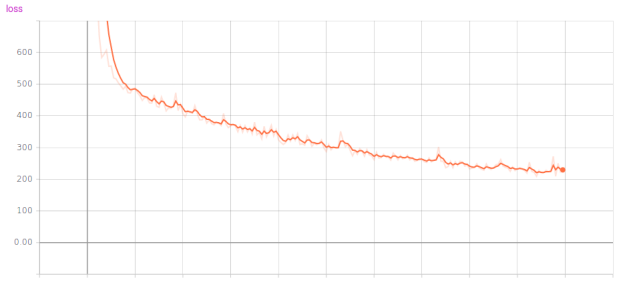

Figure 14. Valor da função da perda durante o treinamento para 200 épocas.

significativa, também pode-se observar que não ocorreu *overfitting* no modelo. Na figura 15, pode-se observar algumas previsões feitas com o modelo, mostrando que todas as previsões chegaram próximas do resultado real e não havendo grandes diferenças.

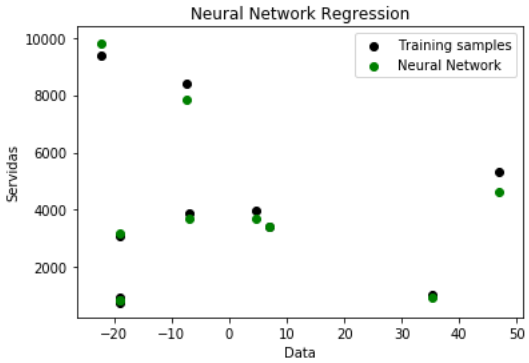

Figure 15. Alguns resultados obtidos para o modelo Rede Neural com tres camada ocultas.

K. Nosso Melhor Modelo

Realizando uma avaliação dos modelos estudados, pode-se observar que os modelos de V-E, V-F, V-G e V-J tiveram um empate técnico. Já que apesar da rede neural obter o melhor resultado, foi por uma mínima diferença. Decidiu-se então testar os dados de teste no modelo das Redes Neurais. Os resultados finais obtidos estão apresentados na tabela XI, onde pode-se ver que nosso modelo não apresentou *overfitting*.

Table XI
Resultados finais do projeto, fazendo uso do conjunto de teste com o modelo Rede Neurais.

Mean Square Error	R2 Score
440.20	90.67%

VI. Conclusões e Trabalhos Futuros

Neste trabalho foi realizado um estudo de diversos algoritmos de Aprendizado de Máquina para reduzir o desperdício nos restaurante universitários da UNICAMP. A principal dificuldade da equipe ao trabalhar com dados

reais foi a etapa de preprocessamento do conjunto dados, pois alguns dados necessitavam de limpeza, normalização, entre outras transformações. Além do fato de que adicionamos mais features ao conjunto original como período de férias e variáveis climáticas, na busca de obter um modelo que melhor explicasse o nosso problema.Utilizamos também análise de dados para visualização dos dados, podendo reconhecer alguns padrões e verificar as variáveis mais importantes como Tipo de Prato e Período de Férias. O nosso melhor modelo obtido foi com Rede Neurais (90.67%) garantindo um baixo erro médio quadrático, sendo melhores que os valores estimados empiricamente no atual contexto. Apesar da Rede Neural ter obtido o melhor resultado, outros modelos também obtiveram excelentes resultados como a Random Forest.

Como trabalhos futuros:

- Desenvolver uma ferramenta com nosso melhor modelo para ajudar a planejar o número de pratos (demanda diária), reduzindo assim o desperdício nos restaurantes da UNICAMP;
- Considerando que no restaurante do UNICAMP é preferível planejar pratos para que sobrem e não para que faltem, na implementação de melhores modelos poderiam ser consideradas penalizações quando os modelos predizem por demandas inferiores as do *groundtruth* (demanda real);
- Apesar do melhor resultado obtido ter sido com Rede Neurais, poder-se-ia continuar trabalhando com os modelos de *Random Forest*, já que eles permitem saber quais tipos de *features* são mais relevantes, como se esta fazendo a decisão do resultado, entre outras coisas.
- Obter mais dados para que seja possível a aplicação de modelos mais sofisticados como Redes Neurais Recorrentes.

VII. Agradecimentos

Agradecemos à Prefeitura Universitária e ao CEPAGRI pelo fornecimento dos dados utilizados neste trabalho. Em especial, agradecemos à Professora Dra. Sandra Avila pelo apoio e orientação nos estudos.

References

[1] Giachero, Augusto F. and Cruz, Guilherme R. and Rodrigues, Henrique H. and Pereira, Hugo R. **Análise do Fluxo de Pessoas e Estimativa de Público por meio de Redes Neurais nos Restaurantes Universitários**. UNICAMP. Universidade Estadual de Campinas (UNICAMP). São Paulo, 2016.

[2] Rocha, José Celso, Felipe Delestro Matos, and Fernando Frei. "Utilização de redes neurais artificiais para a determinação do número de refeições diárias de um restaurante universitário." Revista de Nutrição (2011): 735-742.

[3] Equipe eCycle. Desperdício de alimentos: causas e prejuízos econômicos e ambientais. **eCycle**, 2018. Disponível em: https://www.ecycle.com.br/component/content/article/62-alimentos/3007-desperdicio-de-alimentos-quais-sao-as-causas-e-os-prejuizos-economicos-e-ambientais-desse-problema.html . Acesso em: 21 de nov. 2018

[4] Top 6 Errors: Novice Machine Learning Engineers Make, Christopher Dossman, 2015. Disponível em: https://medium.com/ai%C2%B3-theory-practice-business/top-6-errors-novice-machine-learning-engineers-make-e82273d394db. Acesso em: 6 de nov. de 2018.

[5] Qual a diferença entre LASSO e Ridge Regression?, Flávio Clésio, 2015. Disponível em: https://mineracaodedados.wordpress.com/2015/06/20/qual-a-diferenca-entre-lasso-e-ridge-regression/. Acesso em: 15 de nov. de 2018.

[6] M. Bentlemsan and E. Zemouri and D. Bouchaffra and B. Yahya-Zoubir and K. Ferroudji. Random Forest and Filter Bank Common Spatial Patterns for EEG-Based Motor Imagery Classification in 5th International Conference on Intelligent Systems, Modelling and Simulation, p. 235-238, jan- 2014.

[7] Kavitha S, Varuna S and Ramya R, "A comparative analysis on linear regression and support vector regression," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-5.

[8] X. MA, Y. TIAN, C. LUO and Y. ZHANG, "Predicting Future Visitors Of Restaurants Using Big Data," 2018 International Conference on Machine Learning and Cybernetics (ICMLC), Chengdu, 2018, pp. 269-274.

[9] CHAVES, Bruno Butilhão. Estudo do algoritmo AdaBoost de aprendizagem de máquina aplicado a sensores e sistemas embarcados. 2011. Dissertação (Mestrado em Engenharia de Controle e Automação Mecânica) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2011.