C964 Computer Science Capstone

Western Governors University

Amy Saunders
Student ID #000686585

Signature: *Amy Saunders*

Date: 08/29/2021

Table of Contents

Letter of Transmittal

To: Viking University Executives

From: ASaunders@EducationSolutions.com

Date: 28 August, 2021

Subject: Student performance predictions

To whom it may concern,

Higher education is becoming more and more competitive and, as a result, students are becoming more selective and demanding. Educational institutions must respond by providing better educational opportunities.

The ability to predict student performance is essential to higher learning institutions for a couple of reasons. First, student success is often used as a metric for the institution's performance. Universities with higher performing students are rated higher, attracting more and brighter students and more funding. Second, the ability to predict student performance allows learning institutions to provide personalized resources to the students who need it most, improving learning experiences overall.

More benefits of accurate student performance predictions:

- Early detection of students at risk allows educators to take preventative measures.
- Students can be more accurately placed in classes that will challenge them and keep them interested without overwhelming them.
- It can help teachers improve their skills.
- It can help decision-makers make fact-based decisions.
- Resource allocation plans can be fact-driven.
- Students can track their own academic progress.
- Administrators can be aware of drop-out tendencies among students which will help assure retention.

Here at Education Solutions, we specialize in being able to predict educational outcomes. We work hand in hand with educators like you to bring the benefits of student performance prediction to the education industry.

We know you want to provide your students with the very best education possible and the ability to accurately predict student performance will help you to be able to do just that. Further, as your

students improve and become more successful, your institution will benefit from an improved reputation and your funding will be increased, thus improving your ability to serve your students even further.

Typical applications like this take about 6 weeks to develop and cost about $56,000, depending on the size of the development team, how polished you want the final interface and other factors. We would love to sit down with you and show you some examples of our work, talk about how we can improve your educational process, and design a project roadmap for you that will give you a more detailed time and cost estimate.

Our excellent team of developers, programmers, and engineers has many years of experience developing web & mobile apps for all types of industries. They will work closely with you to develop exactly what you need. Once we've built your custom application, we hand over all of the proprietary code and intellectual property rights so you own the application. We ensure that every app we build is scalable, intuitive, user-friendly and secure. If you have any questions, please feel free to reach out. I look forward to hearing from you.

Sincerely,

Amy Saunders

Education Solutions

A. Project Recommendation

Education Solutions proposes a Performance Prediction application for Viking University that will allow administrators, educators and students alike to accurately predict student performance based on data gathered from the student.

A.1. Problem Summary

Modern universities are competing with other universities not only for students, but also for grants and research money. In addition, they are constantly being ranked against other universities based on student performance, among other factors. It is in their best interest to admit and retain the highest quality students possible. Predicting student performance can be difficult just because of the sheer amount of student data available. It is difficult to determine which data about the student factors most heavily into the predictions.

This Student Performance prediction app is designed to do all of that for you. Students can input their own information such as previous grades and parents' education levels to see their predicted measure of success, or educational institutions can use it to determine which students are most likely to perform well and thus benefit the institution. The application does not include support measures such as remediation plans. It strictly evaluates the student data and returns a number from 0 - 20, with 20 representing the pinnacle of student performance.

A.2. Application Benefits

Our Student Performance Prediction application will benefit Viking University administrators by using admissions criteria, such as past academic performance and various demographic data, to measure and predict student performance. It can help them to both identify and serve at-risk students, assuring retention, and also to better allocate resources where they are most needed. From the educators' perspective, predicting student performance can help them be proactive in guiding students towards areas that need improvement. Students will benefit from a better education, resulting from the ability of the university to assemble a streamlined, individualized plan to accomplish learning most efficiently and effectively. Students will also benefit from being able to see, at-a-glance, exactly which parameters of their lives most effectively improve their own performance prediction so they can make necessary changes. For example, as the slider on the app that represents the amount of time the student spends studying is increased or decreased, the student sees the direct effect that study time has on performance.

A.3. Application Description

The Student Performance Prediction application will ask students (or educators or administrators) a number of questions which will be used to predict student performance in the future. The application will return the prediction as a number, 0-20, with 20 being the highest, along with a series of visualizations comparing the different answers to the prediction. For example, two questions ask about the student's parents' educational levels, which will be compared to the predicted performance of the student in a bar graph. A scatterplot will compare

whether and how much alcohol the student consumes to his predicted performance. And a line graph will compare study hours to student performance. That way all interested parties in the student's education can see easily and intuitively which factors in his life are the biggest hindrances or advantages to excellent performance.

A.4. Data Description

I will use a dataset from Kaggle, originally collected from two Portuguese schools in order to train the machine to be able to make accurate predictions. The original dataset includes 33 columns total. The final column is the target (the final grade of the student) and the other 32 columns are the features. Just looking at many of the features, such as gender, age and address leads me to believe they would not correlate at all with the target or contribute to the accuracy of the predictions. Other features, such as whether and how often the student drank alcohol, the time the student spent studying outside of class and whether the student planned to attend college seem like they would correlate highly. So I will use SciKitLearn to put all of the features and target columns into a correlation matrix (heatmap) in order to see at a glance which columns correlate and how well. From there, I will eliminate columns that don't correlate.

My next step will be to clean the data, making sure there are no empty or missing values in the dataset. I will either fill in or eliminate rows with missing data. From there, I will need to make sure all of the columns I'm using contain strictly numerical data. For example, column 'higher' contains binary data (Yes/No) about whether the student plans to attend university. I will replace Yes with 1 and No with 0. Then the data will be ready to use in whichever regression models we choose to try. We are using regression models because the dataset we are using presents the very most correlation features, G1 and G2, representing the grades in periods 1 and 2, as numbers, 0 - 20.

A.5. Objective and Hypotheses

The primary objective of this application is to predict educational outcomes based on various features of students by building models based on data that is collected from learning management systems and registration data. Moreover, we use Visualization approaches to gain a better understanding of the various features of educational data sets. The hypothesis is that we can predict student performance to within about 10% accuracy. If we can achieve accuracy to within 10% error in making our student performance predictions then we will proceed with the commercialization of the app.

A.6. Methodology

We will use the AGILE methodology for this project because it is fast, flexible, adaptable and error-resistant. Continuous testing and feature integration are a key factor to the success of this application. We will release the app in two separate deployment phases, first to university administrators, with review, making sure that the software is compatible with their other tools and is bug-free, then to a small group of students for testing before being officially deployed. We

will focus on the lean principles of: eliminating waste, building quality, fast delivery and optimization so we can build exactly what you need while remaining within both time and price constraints.

A.7. Funding Requirements

All of the software and tools required to build the Student Performance Predictor app, including Anaconda tools, Jupyter Notebook, Python, Voila and various packages and libraries, are open source and therefore will not incur any cost. Development hours, hardware, server space and employee salaries will cost $56,000, paid in two installments of $28,000 each, prior to each phase of development.

A.8. Stakeholders Impact

One benefit of applying Learning Analytics, Educational Data Mining techniques to predict student performance is that it enhances the understanding of the educational process by the various stakeholders: students, educators, and administrators. As understanding of the educational process improves, administrators are enabled to make small but effective improvements incrementally, checking progress repeatedly, so they can gradually assure that the institution is on a good trajectory toward the goals and objectives they have set for themselves. As they help to provide feedback, instructors are empowered to participate in the process and can see positive outcomes from their work. Students benefit from the process as their educational outcomes are improved.

A.9. Data Precautions

Modern universities use Learning Management Systems such as Canvas to present coursework, track scores, provide communication between instructors and students and more. Registration data, such as socioeconomic information, ethnicity and family background provides even further data. Machine Learning algorithms can be applied to all of this data to derive insightful information. All of these features can potentially affect students' performance and make excellent predictors. However, most of this information is considered private, the United States has limited their use by passing the Family Educational Rights and Privacy Act (FERPA).

FERPA is a federal privacy law that gives parents certain protections with regard to their children's education records, such as disciplinary records, report cards, transcripts, contact information, demographic information and class schedules. Students age 18 and up are protected themselves, and even parents cannot access their information. Educational institutions must separate and keep private all identifying information from datasets in order to protect students' identities.

A.10. Developer's Expertise

This development team will be comprised of two junior developers, led by a senior developer acting as the product owner. We will have a senior software engineer onsite to act as a scrum master. Our excellent team of developers and engineers has many years of experience in data analytics, machine learning and has developed web & mobile apps for many different industries.

Section B
Project Proposal for IT Professionals

B.1. Problem Statement

Educational institutions have had to make significant changes to keep pace with technology. From online courses to MOOC's to implementing various learning tools and system-wide Learning Management Systems, universities are going to great lengths to improve student education and accessibility. Most of the tools make life easier for educators and students, allowing the faculty to distribute, grade and track the assignments given to the students and allowing students to keep track of assignments and their deadlines.

Recently, schools have begun applying machine learning algorithms to student data, gathered from Learning Management Systems and registration in order to help them determine how to best serve their students. Another way to use this data is to see which factors most significantly impact student performance, train a model and use it to predict student performance. The ability to predict student performance will allow educational institutions to get out in front of problems before they arise, to help alleviate drop-out and to provide both students and educators a better learning and teaching experience.

B.2. Customer Summary

The customer base will include all of Viking University's current and future students, administrators and educators. The Student Performance Prediction app will fit customer needs by:

1. Administrators need a tool that is intuitive, user-friendly and quick to be able to determine whether students will be an asset to the university and whether they are prepared for certain courses or in need of remediation.
2. Educators also need a simple, intuitive tool to help them assess the trajectory of a students skill set and understanding to determine best placement for students and to help them determine whether or when remediation is needed and how to best achieve it.
3. Students will be able to use the same app to see for themselves whether they are adequately prepared for the next course they need to take based on their likelihood of success.

B.3. Existing System Analysis

Viking University already has many tools in place within their Learning Management System. They already have in-house servers and host their own database and website. The Student Performance Predictor application we plan to build will integrate with the university database and website. All of the software tools that we will use, such as Python, Jupyter Notebook, Voila and various libraries, will be open-source and publicly available. Because it is written in Python it will be easily adaptable for any environment.

B.4. Data

We will use a dataset from Kaggle for our training data with which to model the app. We already have a specific dataset in mind. It is currently stored in a single csv file with 33 columns full of demographic and educational data pertaining to a group of students from Malaysia. We have chosen to use the csv format for a dataset because it is small, easy to parse and easy to use in Jupyter Notebook. It is also the most common way of storing data of this type.

We will have to clean the data to make it usable, meaning we must fill in missing values or eliminate rows with missing values. The dataset also has a bunch of zeros in critical places, which seem to have been used to replace missing data. Those zeros might skew our results and decrease our overall accuracy, so we may replace them with mean values instead, or we may remove the rows with zeroes in critical places entirely. There don't seem to be any gross outliers. When using a regression model to train and test a machine, the dataset features should be converted to numerical values. We will convert all binary values like Yes/No to ones and zeros. The columns that have string data types, like name and address, we don't plan to use anyway because they don't correlate with the target column.

B.5. Project Methodology

As discussed above, we will be using the AGILE methodology. We prefer this method because it allows for rapid revision, constant debugging and iterative improvement. It will allow us to complete this project quickly and to the highest standards. The work process will happen as follows:

Planning:

- Our application must provide an accuracy range on all datasets that is within our acceptable limit.
- We will take all customer feedback into consideration and adjust as needed in order to provide a finished product with which customers are satisfied.
- Stakeholders considerations will be discussed and may be added to the change board.

Development:

- The project will be deployed in two phases. The first phase will take about four weeks and the second phase will take about two weeks, for a total of six weeks and approximately 960 employee hours.
- The first phase will consist of development of the prototype and deployment strictly to administrators and educators.

- The second phase will consist of troubleshooting and fixing bugs found during the initial deployment and updating features to the customers satisfaction.

Testing:
- Our first round of black box testing takes place when we send university executives the demo version of our Student Performance Predictor. They will provide feedback we will work off of.
- The first phase of development will consist of several rounds of unit and integration testing. We will allocate two weeks for white box testing.
- The second phase will focus on unit testing based on data collected from prior testing.
- Acceptance testing will take place following the final deployment.

Release:
- The initial deployment will follow the first phase of development.
- The project will be deemed complete after executives have signed off following the second deployment.

Feedback:
- Feedback collected through the first phase will be incorporated into the second phase of development.
- All three groups of stakeholders, students, educators and administrators, will provide feedback at various points during development.

B.6. Project Outcomes

Deliverables will be delivered weekly in scrum meetings.
Project Deliverables:
- Schedule
- Test plans
- Dashboard wireframe
- Runnable code segments

Product Deliverables:
- Fully functioning web application with dashboard
- Database modifications
- Complete source code controlled in GitHub for optimal flexibility
- Training video and manual

B.7. Implementation Plan

The implementation plan will begin with the initial deployment of the prototype to Viking administrators and educators. This release will consist of initial user testing, feature suggestions or additional requirements and identification of bugs. Following this release the application will return to development for the addition of features and to eliminate any bugs identified.

Once the revisions have been completed, the application and other associated resources (a training video and manual) will be rolled out to the students. These resources comprise a brief training video series and two weeks of 24/7 support.

B.8. Evaluation Plan

The evaluation plan for our Student Performance Predictor will consist of measurement satisfaction and functional acceptance. Measurement satisfaction depends on a metric representing the accuracy of the final prediction of student performance. In a regression situation like ours, this means testing the mean absolute error and mean squared error. Both metrics measure the mean departure of the predicted value from the actual value when using training and test datasets.

Functional acceptance depends on the application being able to perform the tasks specified in the requirements, including being able to predict student performance to a certain accuracy and presenting visualizations of features to educate the user about which factors in his life are contributing most negatively or positively to that prediction.

B.9. Resources and Costs

Programming Environment:
Python, Jupyter Notebook and all of the various software libraries we will use are open source and free to use. Viking University already has their own servers on which they host their website and database. We will make some database modifications in conjunction with their IT team, but none of that will cost the client anything in terms of hardware or software.

Environment Costs:
There will be no environment costs.

Human Resource Requirements:
This project is expected to take six weeks to complete and use 2 junior developers, 1 senior developer as the product owner and 1 senior developer as the scrum master.
Junior Developers $35/hr
Senior Developer $65/hr
Scrum Master $75/hr

	Hours	Cost
2 Jr. Developers	40*6*2 = 480	16, 800
1 Sr. Developer	40*6 = 240	15,600
Scrum Master	40*6 = 240	18,000
Reserve		10% = 5040

Total		55,440

B.10. Timeline and Milestones

Event	Start Date	End Date	Developer Hours Required	Dependencies	Assigned Resources
Project begins	10/04/2021	10/04/2021	0	None	Project owner, stakeholders
Black Box testing w/ demo	10/04/2021	10/04/2021	24	None	Project owner, stakeholders
Phase One Development	10/06/2021	11/01/2021	420	Project start	Scrum master, project owner, developers
Application deployment to admins	11/01/2021	11/03/2021	48	Phase one development	Scrum master, project owner, developers, stakeholders
Phase Two Development	11/03/2021	11/12/2021	160	First Application deployment	Scrum master, project owner, developers
Feature Update	11/03/2021	11/12/2021	100	First Application deployment	developers
Debugging	11/03/2021	11/12/2021	100	First Application deployment	developers
Application deployment to select students	11/15/2021	11/15/2021	60	Phase two development, feature update and debugging	Project owner, select stakeholders
Final stakeholder	11/15/2021	11/15/2021	48	Phase two deployment	Project owner, scrum

review and sign off					master, administrators

Section D

D.1. Project Purpose

Educational institutions are making great improvements to education with the use of modern technology. New tools, such as Learning Management Systems, make life easier for administrators, educators and students alike. They facilitate faster, better, easier communication between students and educators. They facilitate distribution, grading and tracking of assignments, which is a huge benefit to both students and educators. University administrators, who are tasked with funding the university and upholding its reputation and rankings, are served well by all the data automatically collected through the use of Learning Management Systems and the online registration system, but only if they can put that data to good use.

One way in which universities can put data collected from students to good use is by using it to better serve their students. They can use machine learning to determine which factors most significantly impact student performance, train a model and use it to predict student performance. When they can see that students are failing a particular course at a higher than normal rate, they can make changes to that course. When they see that a particular demographic of students is struggling in a certain area, they can get out in front of the problem and even potentially reduce the drop-out rate. They can even use the data to pinpoint specific areas in which students are struggling and provide targeted remediation. Another way universities can use student performance prediction to their advantage is during the admissions process.

Universities are ranked on various factors, but one of the most significant is the success of their students in terms of academic scores and graduation rate. The use of machine learning to predict student performance can help universities to admit only the students who are most likely to perform well. The Student Performance Prediction application is the perfect way for Viking University to put all their data to use for the benefit of their administrators, educators and students.

D.2. Datasets

We obtained the raw dataset from the following link:
https://archive.ics.uci.edu/ml/datasets/student+performance

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	0	1	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10

The dataset included 33 columns, 32 features and 1 target. The feature columns include: school attended, gender, age, address, family size, parents cohabitation status, parents jobs and education status, students guardian, study time per week and more. Obviously, many of those columns, such as gender, address and guardian have no bearing on the target (prediction), so we removed them. The final column, G3, is the student performance score, our target. G2 and G1 represent the student's scores in periods 1 and 2.

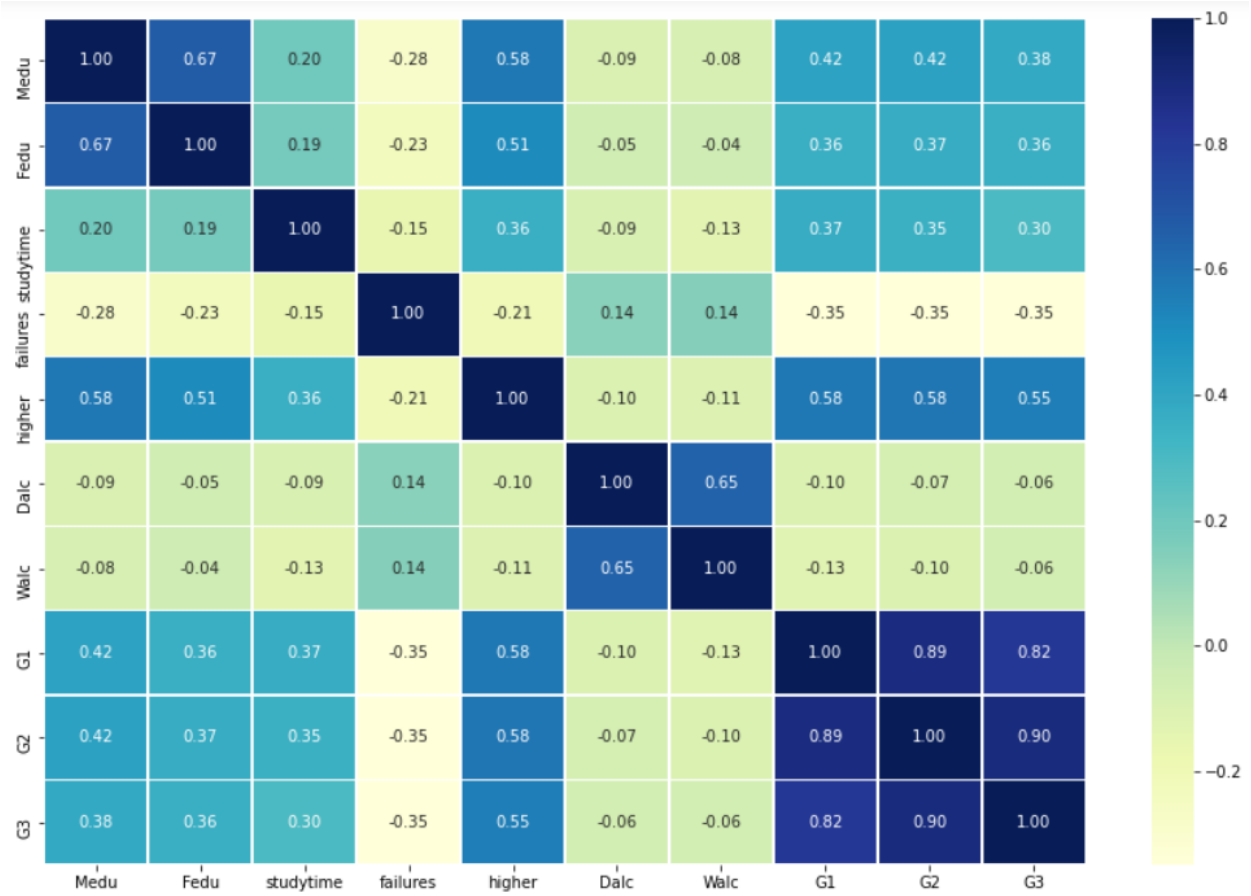
Next, we checked for missing values using df.isna().sum() where df is the DataFrame. This returned the sum of the missing values for each column. Ours returned all zeroes, meaning there was no missing data. However, a quick look at the actual data frame using df.head() and df.tail() showed several zeroes in the G1, G2 and G3 columns, which are our target column and the two columns that seem to correlate most highly with our target column, so they are critical. The zeroes might be indicators of previously missing values and could skew our results, so we decided to run the training with the zeroes in place and then again with the zeroes filled with a mean value, then again with the rows containing zeroes removed entirely. That way we could see which gave us the most accurate result. We used the following command to create a new dataframe with the missing data filled in with a mean value for the column:

df[“column_name"].fillna(dataframe_name["column_name"].mean(), inplace=True)

That result actually gave us a lower accuracy rate than the original data set with the zero values, probably because the mean value of the target column did not correlate with the other columns any better than the original 0 had. We then used the following command to create a new dataframe with the rows containing missing data (zeroes) removed:

```
df = df[df.G2 != 0]
df = df[df.G1 != 0]
df = df[df.G3 != 0]
df = df.reset_index(drop=True)
```

Dropping the rows containing zeroes did improve our accuracy significantly.



After using a correlation matrix to determine which features correlated most highly with our target column, we dropped the features that did not correlate. The highest correlation to student performance(G3) is between the students grades in periods 1 & 2 (G1 & G2). Dalc (weekday alcohol consumption, Walc (weekend alcohol consumption) correlate highly with each other, but only slightly to grades. Medu and Fedu (mother's and father's education levels) and study time correlate positively with grades. Negative correlation was seen between failures, absences and performance. We eliminated the columns that did not correlate with performance and created a new, smaller dataframe in which only one column was non-numeric.

Finally, we used df.info() to determine the data type of each column. Any that were not numeric needed to be changed to numeric in order for our regression model to use them. Higher, which represents whether the student wants to attend university or not, used binary yes/no. We used the following statement to change those to yes = 1 and no = 0:

df.replace(('yes', 'no'), (1, 0), inplace=True)

Without actually changing the dataset, we created a new dataframe with no missing values, mostly correlated features and all-numeric data with which to train our model.

	Medu	Fedu	studytime	failures	higher	Dalc	Walc	G1	G2	G3
0	0	1	1	0	0	1	1	5	6	6
1	1	1	1	0	0	1	1	5	5	6
2	1	1	2	3	1	2	3	7	8	10
3	4	2	3	0	1	1	1	15	14	15
4	3	3	1	0	1	1	2	6	10	10

D.3. Data Product Code

First, we had to split the data we had into two sets, a training set and a test set. We use a training set to train our model, then we use the test set to test it on data it has not seen yet so we can see how well it is trained. We split our data into 80% training data and 20% test data.

Modelling

The highest correlation is between G1, G2 and G3 so we should weight those features most heavily as we consider hyperparameters. We'll used the smaller dataframe from which we eliminated the least helpful columns. And we need to convert "higher" to numerical.

In [23]:
```
# Split data into x/y
# X will be all columns EXCEPT target, so just drop target column
X = reduced_df.drop("G3", axis=1)
y = reduced_df["G3"]   # binary classification

# Split data into train/test sets
np.random.seed(42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [24]:
```
X_train
```

Out[24]:

	Medu	Fedu	studytime	failures	higher	Dalc	Walc	G1	G2
181	3	3	2	0	1	1	2	12	13
194	2	3	1	0	1	1	1	13	14
173	1	2	2	3	0	1	1	8	7
63	2	3	3	0	0	2	4	10	9
253	2	1	1	0	0	1	3	8	9
...	...	...	...	...	...	...	...	...	...
71	1	2	1	0	0	1	1	10	10
106	2	2	1	0	0	1	1	7	8
270	1	3	2	2	0	3	3	9	9
348	4	3	3	0	1	1	3	13	15
102	4	4	1	0	1	1	1	10	13

316 rows × 9 columns

Next, we wanted to try several different regression models to see which would give us the best results. The easiest way to try several models at once was to write a fit (train) and score (test) function that would do it for us. We checked the SciKit Learn Map for regression model recommendations and used the models it recommended. RandomForestRegression() gave us the best results. The following code provides the predictive functionality within the application:

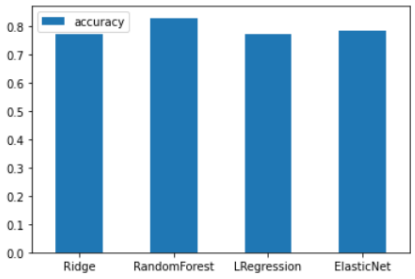
In [26]:
```
# Put models in a dictionary
models = { "Ridge": Ridge(),
           "RandomForest": RandomForestRegressor(),
           "LRegression": LinearRegression(),
           "ElasticNet": ElasticNet()}

# create a function to fit (train) and score models
def fit_and_score(models, X_train, X_test, y_train, y_test):
    """
    Fits and evaluates given machine learning models
    models: a dict of different Scikit-Learn machine learning models
    X_train : training data (no labels/targets)
    X_test: testing data (no labels)
    y_test: testing labels
    y_train: training labels
    """

    #Make a dictionary to keep model scores
    model_scores = {}

    #Loop through models
    for name, model in models.items():
        # Fit the model to the data
        model.fit(X_train, y_train)
        # Evaluate the model and append it's score to model_scores
        # We'll save the name of the model(KNN, LogisticRegressions, RandomForest) to the model_scores empty
        # dictionary as the key and the score as the value
        model_scores[name] = model.score(X_test, y_test)
    return model_scores  # will return a dictionary
```

In [27]:
```
model_scores = fit_and_score(models=models,
                             X_train=X_train,
                             X_test=X_test,
                             y_train=y_train,
                             y_test=y_test)
model_scores
```

It's usually easiest to see results in a visual output, so I created a descriptive bar graph to compare the scores of the various models we tried.

Out[27]:
```
{'Ridge': 0.772164008738353,
 'RandomForest': 0.828101298714787,
 'LRegression': 0.7721207919866366,
 'ElasticNet': 0.7826566472496042}
```

In [28]:
```
model_compare = pd.DataFrame(model_scores, index=["accuracy"])
model_compare.T.plot.bar()
plt.xticks(rotation=0);
```



RandomForest is the best model at ~83%, but not as high as we'd like. Let's try adjusting hyperparameters.

Finally, after determining that RandomForestRegression would give us the best results, we decided to check the mean absolute error and mean squared error, which are evaluation metrics used with regression models. The mean absolute error of a model with respect to a test set is the mean of the absolute values of the individual prediction errors over all of the instances in the test set. The mean squared error tells you how close the regression line is to a set of points.

In [29]:
```
# Let's check MSE and MAE to calculate the error
from sklearn.metrics import mean_absolute_error, mean_squared_error

# calculating the performance of the Random Forest Regression Model
rf = RandomForestRegressor(random_state=42)
rfModel = rf.fit(X_train, y_train)
y_pred_r = rfModel.predict(X_test)
mae_r = mean_absolute_error(y_test, y_pred_r)
mse_r = mean_squared_error(y_test, y_pred_r)
print(f"MAE: {mae_r}  \nMSE: {mse_r}")
```

```
MAE: 1.2781476793248947
MSE: 3.5168857426160334
```

D.4. Hypothesis verification

Our initial hypothesis was that we could predict the performance of a student based on a few parameters such as the amount of time he/she spent studying, the amount of alcohol the student consumes, the number of classes the student has failed in the past and the grades the student received in his/her first and second period classes. After trying several regression models and tuning the hyperparameters of the regression model we found worked best (Random Forest Regression) we tested the model with our test data and found the mean absolute error and mean squared error (the two verification metrics outlined above) verified our hypothesis. Viking University tested the Student Performance Prediction application with some data of their own and found it to be accurate and reliable. The app successfully provides the user with predictive methods.

D.5. Effective Visualizations and Reporting

The visualizations provided in the dashboard are intended to enhance the user experience by providing clearer, more easily understood information. Many people are visual learners and understand information presented in graphs and plots much better than the same information presented in paragraphs. Plotly enables the user to zoom in on certain portions of the graphs, download any of the images and allows for an interactive experience.

We selected our visualizations for the dashboard based on features we felt would be most informative and helpful to students, educators and administrators. For example, we chose to display a scatterplot of student performance based on alcohol consumption and a histogram of student performance based on the amount of time spent studying. We chose these features, t alcohol consumption and study time, because students have control over those factors and we wanted them to see how heavily they factored into performance. Students could easily scroll back up to the input form and reduce the alcohol consumption and increase the study time to see exactly how those changes would impact performance, and it might encourage them to make personal improvements along those lines.

D.6. Accuracy analysis

When building a predictive application like the Student Performance Predictor, your first step is exploratory data analysis (EDA). Not all of the data in the provided dataset will correlate well with the target column so you have to determine which features do correlate and how well. Once you've determined which features correlate well, whether positively or negatively, you use those features to build, train and test your model, then tune the hyperparameters, so that you end up with a model which can accurately predict targets based on highly correlated features.

We followed those steps and built a model that seemed pretty effective at providing predictive outcomes. Our next step was to quantify exactly how effective. Because we used a regression model, the evaluation metrics mean absolute error and mean squared error made the mode sense. Mean squared error tells you how close the regression line is to a set of points, while mean absolute error is the average of the absolute differences between predictions and actual values. It gives you an idea of how wrong your model's predictions are. Mean squared error for our model was ~3.5 and mean absolute error was ~1.3, which are both great results for a regression model. Here is a visualization for better understanding.

In [47]:
```
# Let's create a visualization to see if it helps us understand.
df1 = pd.DataFrame(data={"actual values": y_test, "predicted values": y_pred_r})
df1["differences"] = df1["predicted values"] - df1["actual values"]
df1
```

Out[47]:

	actual values	predicted values	differences
78	10	7.086000	-2.914000
371	12	12.170000	0.170000
248	5	3.960000	-1.040000
55	10	9.526000	-0.474000
390	9	8.372000	-0.628000
...	...	...	...
364	12	11.314133	-0.685867
82	6	4.888000	-1.112000
114	9	8.062000	-0.938000
3	15	14.710000	-0.290000
18	5	1.954000	-3.046000

79 rows × 3 columns

D.7. Application Testing

- Unit Testing occurred extensively throughout the applications development. It consisted of: processing various iterations of the dataset, verifying projected output against actual output, verifying that visualizations displayed as expected and verifying the application performed as expected on various operating systems and web browsers.
- Regression Testing was implemented after bugs were reported or features were added. The purpose of the regression tests was to validate the unit test still functioned as expected alongside the updates.
- System Testing took place on the complete, fully integrated application once it was built and was conducted by the testing team.

- Acceptance Testing was conducted by the administrators to determine whether the requirements of their specifications had been met.

D.8. Application Files

The Student Performance Predictor application was built in Jupyter Notebook and is hosted on Binder. Use the link below to launch the application on Binder in your web browser, without any required installations. That is by far the easiest method for running the application, although it does take a few minutes to load. However, if you want to run the application from the raw source code, you will need the following:

- Jupyter Notebooks
- SciKit Learn
- NumPy
- Ipywidgets
- Pandas
- Matplotlib
- Plotly
- TensorFlow
- Keras
- Voila
- Github
- Binder

LINK TO STUDENT PERFORMANCE PREDICTOR APPLICATION

All of the required files to view and run the initial application demo can also be found here: https://github.com/amymaries/student_performance_predictor. These files will also be submitted with the required documentation. Here are descriptions of each file in the repository:

student-performance-prediction.ipynb - This is the Jupyter Notebook running at the Binder link above. It includes all of the functions used to run the application, but none of the code used to explore the data initially.

math-success-classification1.ipynb - This is my original Jupyter Notebook with all of the notes and the research that went into building the application. It includes the visualizations I used to conduct exploratory data analysis, to determine the correlation of different features, to try various regression models, to tune the hyperparameters and then use cross validation. I copied this notebook, renamed it student-performance-prediction (the file above) and removed most of the investigative material to build the application. That way I didn't have to use tags to hide all of the extraneous information I didn't want showing up in the finished app. I also figured it would run faster.

student-mat.csv - is the dataset used to train and test the model.

Requirements.txt - is the version of each python package used to create the notebook. Binder uses this file to build the interactive application.

D.9. User's Guide

When using the Student Performance Predictor application, please follow these instructions:

1. Use this link to navigate to the application in your web browser.
2. Input values into all of the fields in the widgets, lines 121 - 129.
3. Run the application.
4. Use the provided tools at the end of the notebook to explore the effects of possible changes to your parameters. For example, increasing the amount of study time using the slider should increase student performance in the adjacent scatterplot since they are positively correlated. Or decreasing the amount of alcohol consumption should improve student performance since they are negatively correlated.

D.10. Summation of Learning Experience

I had no prior experience with Machine Learning, aside from the Introduction to Artificial Intelligence class. I did have some Python experience, however, which helped. Before starting this project I took an excellent Machine Learning class on Udemy, where I learned to use Jupyter Notebook, SciKit Learn, Matplotlib, Pandas, NumPy, Google Colab and various Python libraries. I also used Stack Overflow and YouTube tutorials to learn how to build a dashboard GUI for my Jupyter Notebook and to learn how to perform various functions as I'm not yet even remotely fluent in Python.

I spoke with my CI, Dr. Barnhart, several times about the general idea of the project and met with Amy Antonucci for help using Voila and Binder/Heroku to create and host the dashboard and to make it look the way I envisioned it in my head. In my opinion, the Computer Science field is so vast and rapidly growing and changing that it is impossible for any one person to ever master it. Software engineers, machine learning engineers and everyone else who works in the field must also be continually growing and learning. In that respect, this degree has thoroughly prepared me for a future career because I've gotten proficient at reading the stack trace and finding my errors, searching for answers online, constantly learning and asking for help. I've always been a lifelong learner. It's my MO. But I'm used to reading entire sections of the library until I master a subject and then moving on out of boredom. I don't think that will ever be the case working in this field.

Section E

I did not use any sources for this paper.