March 31, 2025

C964: Computer Science Capstone

Task 2 parts A, B, C and D

Part A: Letter of Transmittal

03/31/2025

Dr. John Gretzky

Salve Health Insurance

155 Sawmill Way

Jefferson, WI, 53059

Dear Dr. Gretzky,

I am writing this proposal to present a developed solution to Salve's premium cost development and plan tier optimization issues. As you are aware, the company has been struggling to develop a proper plan tier and premium cost assignment method for new clients. The company has consistently lost money over the past fiscal year due to unpredicted medical charges from our clients. A better method of setting premium and coverage rates is needed, or the company will continue to lose money.

My solution will solve this problem by creating a system that accurately predicts client healthcare charges. The application will analyze relevant demographic data from patients nationwide, including their incurred medical charges. The application can chart these demographics and give insights into the effect each demographic has on the overall charges incurred by the person. Most importantly, the application will use this data to build and train a prediction model. This model will then be able to take the demographic data of potential clients as input and give estimated medical expenses as output. This will provide a baseline to establish plan tiers and costs for prospective clients.

The application will benefit the company in multiple ways. First, the data insights will provide context for what customer demographics might incur more risk for the company. This will help to establish limitations for certain plan types based on high-risk demographics. Second, the application will provide accurate cost estimates and enable our company to establish proper premiums, coverage rates, and deductibles. This will effectively solve the problem of overspending by allowing the company to shift costs to high-risk clients better. In addition to the financial benefits, the application will easily scale to learn from new data over time. This means that the application can learn from our current clients and adjust its accuracy over time. This will enable the company to continue balancing costs, premiums, and plan tiers each year based on the previous year's data.

A basic form of the application will be simple to develop using an open-source dataset, but it would be best to set aside money to acquire high-quality, relevant data over time. We will also require a separate server to host this application outside our standard website servers. Additional developers should be hired to assist the I.T. department and me in building, testing, and deploying the model. The costs rounded up will amount to roughly $100,000 in personnel costs, $15,000 in new hardware, $2,000 in software licensing, and $5,000 in data acquisition. This would amount to around $122,000 total.

The ordered timeline of the project would be to acquire relevant data, clean and format the data, explore and analyze the data, build the application interface for visual analysis of the data, choose and design the appropriate prediction model, build the model, prepare, format, and split data for training the model, train the model, test and fine-tune the model, validate the model, create a user interface to interact with the model, and to deploy the application to a server for company use. The estimated timeframe would be 4 weeks for gathering, cleaning, and analyzing the data into a dataset, 2 weeks for designing the application interface for visualizing the data, 2 weeks of choosing and designing a predictive model, 4 weeks of preparing data, building, and training the model, 4 weeks of fine-tuning and validating the model, 2 weeks finishing the user interface, and 2 weeks of deployment. This amounts to roughly 20 weeks of development time.

The data for the initial model will be gathered from an open-source database containing medical charges and demographic data of people across the nation. This data will contain no PII (Personal Identifiable Information) and, as such, will not pose any security risks. This data will suffice to train and build the initial model, but the model accuracy can be improved by adding relevant data over time. Much of this data could be sourced from our own customer records. Proper precautions will be taken when acquiring this data. The data should be scrubbed of any PII before being transferred and used in the model. The model should never have access to sensitive customer data. Proper access controls have already been implemented in our customer database, and all regulations and security protocols will be followed when processing this data.

I plan to lead development on this project personally. I have been programming for 15 years and have recently acquired a Computer Science degree. The many projects I have worked on include tech demos, statistical tools, websites (including our company site), embedded systems (hearing aids), and web-scraping analytics platforms. I have extensive experience working with statistical tools and machine learning models (what we use for prediction systems). As such, I am confident I can complete this project along with our I.T. team and some additional temporary hires.

I am certain that this application will aid Salve Insurance in cutting costs and becoming profitable once more. If you are still unsure, I can provide a small demonstration of this prediction model and how useful it can be. Please let me know what you think of this proposal when you get the chance. I look forward to hearing from you.

Sincerely,

[illegible]

Josiah Fesh, Software Developer

Part B: Project Proposal Plan

The project proposal should target your client’s middle management. This audience may be IT professionals but have limited computer science expertise. Use appropriate industry jargon and sufficient technical details to describe the proposed project and its application. Remember, you’re establishing the technical context for your project and how it will be implemented for the client. **Write everything in the future tense.**

Project Summary

The primary goal of this project is to solve the problem of predicting healthcare costs. Assessing clients' risk and predicting charges they might accumulate is essential to assigning proper plan premiums, deductibles, and coverage rates. This is the current problem that Salve Health Insurance faces. The company’s current method of assessing risk involves having the client fill out a questionnaire and providing plan options based on their answers. This method partially succeeded initially but has caused considerable losses over the past few years. The company desperately needs a new system to assess patient demographics and predict the cost of coverage.

The project will solve these problems by providing an application to analyze data and provide predictions. The application will include analyses and visualizations of patient data, as well as a linear regression prediction model to take patient demographics and provide a prediction for healthcare charges. This initial application will be contained in an interactive notebook and hosted on a company server. The notebook application will include the aforementioned data analysis tools, a prediction model, and a user interface to interact with the model and get predictions. This project will also involve developing documentation that explains the application notebook's framework, development process, datasets, and inner workings. The documentation will include a user guide on how to install, run, and use the application.

The application will provide a simple interface for analyzing customer demographics and making cost predictions. This will alleviate the company’s issue with assessing risk and assigning costs. With better cost prediction and adjusted plan costs, Salve Health Insurance can cut costs and become profitable again.

Data Summary

The initial data for this project will be sourced from this dataset on Kaggle: https://www.kaggle.com/datasets/willianoliveiragibin/healthcare-insurance.

This dataset contains over 1300 records of demographic data and associated medical charges for patients across the U.S. The demographics include age, sex, BMI, number of children, region, and smoker status. With these data fields, the application can easily find correlations and train a linear regression model for predicting healthcare costs. The dataset will be downloaded as a CSV file and loaded into the application for analysis and building the prediction model.

The data will be analyzed for dirty or missing values. Any records with missing data fields will not be carried over to the final dataset for training the model. Non-numerical data will be re-formatted as a series of binary fields indicating the existence of specific values. This will allow that data to be used to train the linear regression model.

During the development of the model, the data will split into training and testing sets. These sets will be further split by dependent variable (the prediction field) and independent variable (the demographic fields). The training sets will be used to build and train the model, and the test sets will be used to fine-tune and validate the model's performance.

After deployment, the model will be maintained and tuned with additional demographic data from the company's customer database. This data will keep the model accurate over time as the economy changes. With the use of company customer records comes new security measures. The initial dataset from Kaggle contains no PII (Personal Identifiable Information) and needs no cleaning or additional security measures. It is generic health data already available to the public. On the other hand, the company records will contain PII for current clients of Salve Health Insurance. These records come from the company database and must have proper access controls established to bar unauthorized access. The application cannot have direct access to the raw records in the company database. Instead, these records must be copied and scrubbed of any sensitive or identifying information by authorized personnel and fed into a secondary training database for the model. The model can then access these cleansed records, which should only contain generic demographic data, similar to the initial training dataset.

Implementation

The development of this project will follow the Agile methodology of iterative development cycles. The initial cycle will focus on developing a basic analysis application. The application will be able to import and explore the dataset. The application will provide visualizations and insights into the raw dataset.

The second cycle will focus primarily on selecting, cleaning, and reformatting the data fields for training the prediction model. This cycle will rely on the insights and analysis gained from the first development cycle. All fields selected should be transformed to numerical format to be used in a linear regression model, and code will be written to properly split the datasets into training and testing sets.

The third cycle will build and train the prediction model. The model will coded and then fitted to the training sets. After training the model, the rest of the third development cycle will focus on testing the model using the appropriate sets. Various metrics and visualizations will be employed to assess the fit of the model.

The fourth cycle will use metrics from the third development cycle to fine-tune and finalize the prediction model. The model will be analyzed again with new unseen data and tested on the same metrics until the measurements fall within acceptable parameters.

The fifth cycle will develop a user interface for interacting with the model. The initial interface will be simple and allow the user to input demographic information into the model. The model will then output the predicted medical charges in dollars. Proper security will be implemented to sanitize user input and remove accidental system exploits.

The sixth cycle will deploy the model to the company servers. The model will be accessible to authorized individuals, allowing the company to begin generating predictions for prospective customers. This will complete the initial development project.

Additional cycles will maintain the application and create a secondary database to store cleansed customer demographic information from current customers of Salve Health Insurance. The model will access this data to improve and maintain accuracy over time.

Timeline

Milestone or deliverable:	**Duration (hours or days):**	**Projected start date:**	**Anticipated end date:**
Visual Analysis Platform	30 days	05/01/2025	05/31/2025
Split and Formatted Datasets	15 days	06/01/2025	06/15/2025
Prototype Prediction Model	30 days	06/15/2025	07/15/2025
Finished Prediction Model	30 days	07/15/2025	08/15/2025
User Interface	15 days	08/15/2025	08/30/2025
Documentation	15 days	09/01/2025	09/15/2025
Deployment	15 days	09/15/2025	09/30/2025
Long-Term Training Database Setup and Connection	30 days	10/01/2025	10/30/2025

Evaluation Plan

The project will verify that the application performs as expected throughout the development process. This verification process will involve unit tests, integration tests, and user testing to ensure the application performs as intended. User testing and maintenance will continue after deployment to ensure the application continues to support the functions of the company.

The majority of verification will be done through unit testing. A unit test will be created for each new function or piece of logic added to the application. These tests will ensure that the building blocks of our application continue to perform their expected operations. Designated inputs should match predefined outputs, and these tests should be run during every stage of development.

Integration testing will begin soon after the framework of the application is developed. These tests will involve testing separate systems, components, and processes and ensuring they work or “integrate” together. In particular, the dataset should integrate with the application early in development to ensure all data is loading into the application. Later in development, integration tests will be run to ensure that the application can connect to and load data from a training database to improve accuracy over time. Another important integration test will ensure that the user interface interacts with the prediction model properly, allowing users to input demographics and get a prediction output.

User testing will also be implemented throughout the entire development process and long after the model is deployed. User testing will ensure that the application is easy to navigate and performs as expected for the average user. This process will be open to as many users as possible, both within the company and outside hires. The more people using and testing the application, the more bugs will be uncovered and fixed. Proper support staff and bug report channels should be available after deployment to allow users to continue submitting error reports to maintenance. This will increase the lifetime of the application and increase user satisfaction.

The model itself will be validated through the use of common linear regression metrics. These metrics will include Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Root Error (MRE), and R^2 score. The first three metrics measure the margin of the model’s prediction errors. MAE gives the most basic overview, while MSE and MRE are more punishing for large prediction errors. MAE should be within \$5000 for predictions. MRE should be within \$2000 of MAE. The R^2 score will address the model’s overall fit to the data it is working with. The closer to 1, the better the score. We will want a score of 0.75 or higher to ensure our model provides good enough cost predictions.

Resources and Costs

Item:	**Description:**	**Cost Type:**	**Total:**
Server Hardware	Additional local server machine to host application	One-time purchase: $1000	$1000
Temporary Workstations	Two work computers for temporary hires	Rentals: $200/month for 3 months each	$1200
Temporary Hires	Two part-time software developers to assist with the prediction model	Hourly employees: $24/hour each for 4160 hours (3 months)	$100,000 (rounded)
Other Labor	Overtime and bonuses for the in-house team for project work. The base salary is covered in the main company budget.	Estimated totals rounded up	$50,000
Software Licenses	Software licensing for development environment and server operating system	One-time purchase: $1500	$1500
Maintenance	Power bills, database backup hosting, miscellaneous service fees	Monthly Fees: $1000/month	$1000/month
Totals:	**Development Costs + Maintenance Rate**	**One time + monthly**	**$187,000 (spending) + $1000/month (maintenance)**

Part C: Application

Part D: Post-implementation Report

Solution Summary

Salve Health Insurance needed a system to analyze client data and predict insurance costs. This system was to aid the company in assigning plan types and premium charges for clients. The company was struggling to assign premium charges and benefits properly, and they had been losing money. The proposed system was to solve this problem by providing better insights and a predictive model to assess what charges a client might accrue before signing up for a plan.

We developed this application to solve Salve's problem by analyzing generic demographic data from similar regional patients. The application explored the demographics through visualization and other descriptive statistical means. These insights signified important trends and correlations between the demographic data and accrued healthcare charges. The application then built a linear regression model and trained it on a large subset of the data. The model was then tested with the remaining data, providing metrics to determine the model's usefulness. The model has been equipped with an interface to allow Salve to predict possible insurance charges based on client demographics, effectively solving the company's problem and providing a solution for adjusting premiums for potential clients.

Data Summary

Data for this project was sourced from this dataset on Kaggle: https://www.kaggle.com/datasets/willianoliveiragibin/healthcare-insurance.

This dataset provided over 1300 records of patient demographics and healthcare charges for patients in the U.S. The data included demographics of age, sex, BMI, number of children, smoker status, region of country, and healthcare charges.

The data was downloaded as a CSV file and included in the application folder. The raw data was imported from the file into the Python program using the Pandas library read_csv function and stored in a DataFrame object.

The data was then parsed and checked for null values. Once no null values were left, we performed a descriptive analysis of the data, checking the distribution of features and correlation between data fields. After exploring the data, we created a new formatted dataset to use for training the linear regression model. Any non-numerical data was converted through a series of customized function calls to create a series of binary fields to represent the presence of different values. This effectively converted non-numerical data into a numerical representation.

During model training, the data was split into four parts. First, the data was split into an independent data set X (patient demographics) and an independent variable set y (healthcare charges). From there, each of these sets was split once more into training (80%) and testing (20%) sets. The model was trained on the testing set of X and y. Once the model was trained, the model created a prediction set (predicted charges) using the test set of X. This prediction set was then compared with the test set y (the actual charges). This comparison provided metrics to evaluate the model performance and fit.

No sensitive data was included in this data set. The set includes generic, non-identifiable information only. As a result, no additional security measures were needed for this dataset.

Machine Learning

In this application, we employed a linear regression machine learning algorithm to predict healthcare charges from patient demographics. A linear regression algorithm is a statistical tool that can be used to predict an independent variable through linear correlation with a dependent variable (e.g., y = mx + b). In this application, patient demographics provided independent variables (age, sex, BMI, etc.), and the patient's healthcare charges were the dependent variable to be predicted. A linear regression algorithm with multiple dependent variables follows this form:

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \varepsilon$.

- Y – our dependent variable (healthcare charges)
- $X_{1\text{-}n}$ – our independent variables (patient demographics)
- B_0 – our intercept (a base-level starting point for healthcare charges)
- $B_{2\text{-}n}$ – regression coefficients (correlation between demographic and overall charges)
- ε – error term (accounts for unexplained variance in charges)

The model was developed using the SciKitLearn library for Python. The model's development and training process was as follows:

1. The training data was stored in a CSV file and loaded into the application using the Pandas library to read and store the file as a data frame.
2. Data was analyzed, formatted, and selected for use with the model.
    a. Data was analyzed for basic correlation, format, and missing values. No missing values were found. A new dataset was created to store properly formatted data columns.
    b. Non-numerical columns such as 'region' were converted into binary flag fields representing each possible value. These columns were added to the new data set for each record using the original dataset's 'region' field to configure each value. Smoker and sex columns were converted using the same process.

   c. All properly formatted numerical data was selected and placed into the new dataset.
3. Data was split into independent and dependent data (X and y). Data was further split into training (80%) and testing (20%) sets using the "sklearn.train_test_split" function.
4. A linear model was created using the LinearRegression object from sklearn.
5. The model was fitted to the training data (X_train, y_train) using the "fit" method of the linear model. This method trained the model with the training inputs and outputs.
6. The application used the newly trained model to make predictions using the independent variables from the testing data set (X_test). These predictions were saved to a new data set.
7. The final test analyzed predictions against the actual values of test data (predictions vs y_test). This provided an analysis of the model and confirmed how well it worked.

The linear regression algorithm provided a simple but effective solution to predict healthcare charges from patient demographics. This algorithm's greatest strength is its easy interpretability. Even non-technical workers can understand how the model is developed and easily see how each demographic field correlates and contributes to the overall charges. This simplicity also made the model easy to develop and efficient to run. Little development time was needed, and no extensive hardware was needed to run the algorithm. Furthermore, the model easily scales to incorporate new data, requiring little to no changes to train or predict larger datasets. The data can simply be formatted and added to the model, and if the application ever requires a more advanced algorithm (such as decision trees), the linear regression model still provides an excellent performance-analysis benchmark.

Validation

The model was evaluated using SciKitLearn's metric library. Since we used a linear regression model, we used the metrics of mean absolute error, mean squared error, and mean root error to measure error margins.

- Mean Absolute Error (MAE) – The mean difference between actual values and predicted values of test data.
- Mean Squared Error (MSE) – The mean squared difference between actual and predicted values. Punishes outliers more heavily than MAE.
- Mean Root Error (MRE) – The square root of MSE. More useful comparison since it shares the same units as test values.

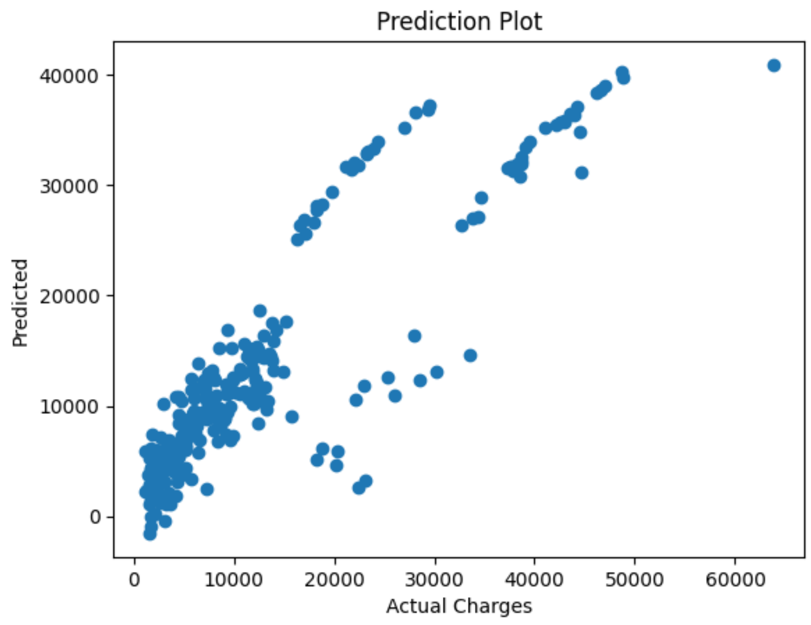
These values indicated the prediction model's margin of error. The actual values were as follows:

- Mean Absolute Error: $4181
- Mean Squared Error: ($\$^2$)33596916

- Mean Root Error: $5796

As shown, the model averaged around $4000 in absolute error margin. Since our data includes data in the tens of thousands, this is accurate enough to provide a rough estimation and fulfill the original purpose of our business app. A model that can predict costs within the closest ten-thousand range will help create evaluation risk tiers for potential insurance clients.

To evaluate how well the model fits our data, we employed two further metrics. The first method was to visualize the “fit” of the model by plotting the predicted values against the test values on a chart. An ideally fitted graph should have data clustered around a diagonal line from bottom left to top right. The data should have a fairly equal correlation above or below this imaginary line. The application generated this chart of the model:



The model displays a slight correlation around the diagonal but is heavily loaded above the line. This indicates that our model predicts higher than the average actual cost.

The second metric we used to evaluate fit was the R^2 score. A model’s R^2 score is a decimal number between 0 and 1, representing how well the data fits the model we are testing. The closer to 1, the better the fit. For our purposes, any score above 0.75 will be acceptable for the application. The score can be calculated using this formula:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

- y_i – The actual values
- $\hat{y}_i$ – The predicted values
- $\bar{y}$ – The mean of the actual values

We ran this test using the "sklearn.metrics.r2_score" function from the SciKitLearn library. The score evaluated to:

- R^2 Score: 0.784

This is within our decided parameters, so the model fits the data well enough for our business purposes. If we decide to improve the model further, we can use insight from the prediction plot chart from before. The chart indicated that predictions were higher than expected on average. This fact, paired with the major difference between MAE and MRE scores from earlier, could indicate that the data used contains major outliers in the charge field. We could create a histogram to explore the distribution of charges among patients and locate such outliers. If massive outliers are found, we could drop the associated records from our training data and possibly create a more accurate model. This is one possibility for future improvement, but it should be explored cautiously so as not to "overfit" the model.

Visualizations

All data charts and visualizations are code-generated and included in the application notebook. The plot chart for the application prediction plot was also included above in the validation section. Screenshots are also included in the user guide to assist with application setup.

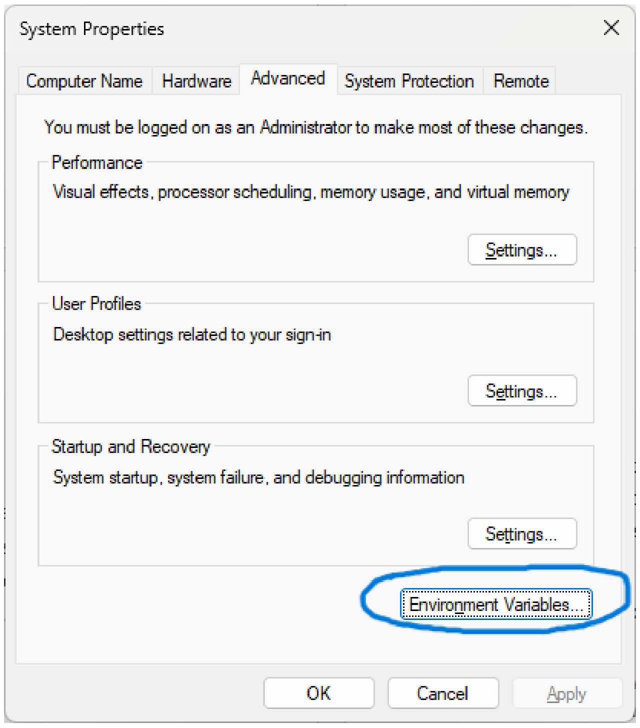
User Guide

Currently, this application is not hosted anywhere and is only available in file format as a jupyter notebook. The files will be sent to the client or can be downloaded from GitHub here. Before natively running this application, several packages must be installed on the user's computer. These packages will be listed below as well as in the readme file included in the application file folder.

Requirements:

- Python 3 – The latest version of Python 3 for Windows can be downloaded from https://www.python.org/downloads/windows/. Use the Windows installer for easy installation. Make sure to select the option to add Python to the system path and make a note of the installation directory, as you might need it later. Default directory is usually something like: C:\Users\(*User's Name*)\AppData\Local\Programs\Python\(*Python Version*)
- Pip – Pip is Python's built-in package manager and is the easiest way to install the remaining dependencies. Pip should be included with your Python installation but will require some configuration to be accessed and run through the command line. Open your command line and try typing "pip --version". If that doesn't work, you will need to

add Python's script folder to the system path. Open your start menu and search for "system environment variables" or something similar. You should find an option to edit them. Opening this should take you to a system properties configuration window like the one pictured below. Select "Environment Variables":



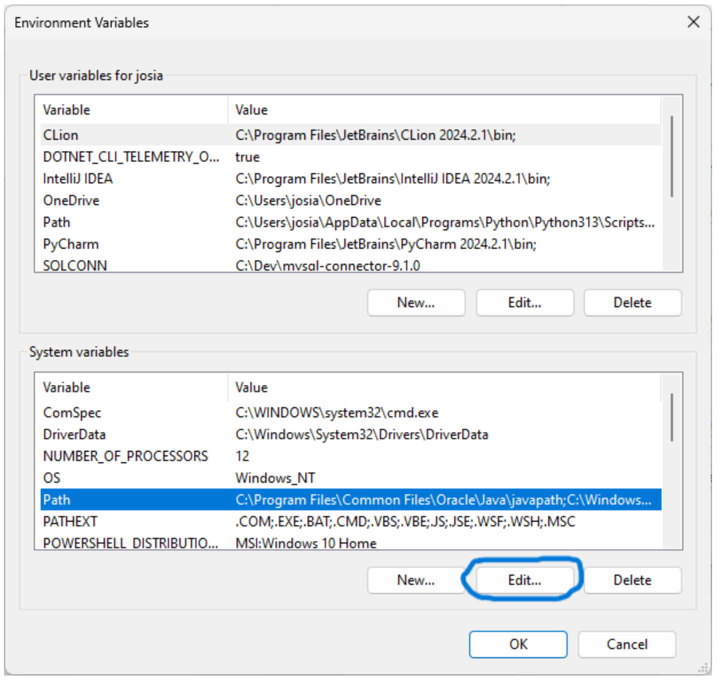
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

This will lead to another window; select the Path variable from system variables and click edit as displayed below:



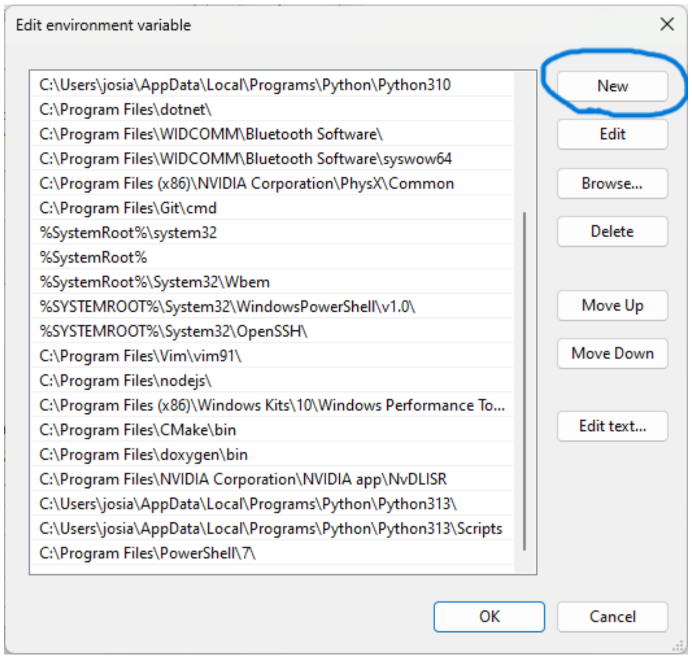
User variables for josia

Variable	Value
CLion	C:\Program Files\JetBrains\CLion 2024.2.1\bin;
DOTNET_CLI_TELEMETRY_O...	true
IntelliJ IDEA	C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\bin;
OneDrive	C:\Users\josia\OneDrive
Path	C:\Users\josia\AppData\Local\Programs\Python\Python313\Scripts...
PyCharm	C:\Program Files\JetBrains\PyCharm 2024.2.1\bin;
SOLCONN	C:\Dev\mysql-connector-9.1.0

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Windows...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
POWERSHELL_DISTRIBUTIO...	MSI:Windows 10 Home

Click on the new button to add a new path to the system path. Enter the path to the Scripts directory located in your Python installation folder. This folder will be located in the directory you made note of earlier. For example: C:\Users\(*User's Name*)\AppData\Local\Programs\Python\(*Python Version*)\Scripts. Apply changes and restart the command line. The "pip –version" command should now run from the command line.



```
C:\Users\josia\AppData\Local\Programs\Python\Python310
C:\Program Files\dotnet\
C:\Program Files\WIDCOMM\Bluetooth Software\
C:\Program Files\WIDCOMM\Bluetooth Software\syswow64
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files\Git\cmd
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Vim\vim91\
C:\Program Files\nodejs\
C:\Program Files (x86)\Windows Kits\10\Windows Performance To...
C:\Program Files\CMake\bin
C:\Program Files\doxygen\bin
C:\Program Files\NVIDIA Corporation\NVIDIA app\NvDLISR
C:\Users\josia\AppData\Local\Programs\Python\Python313\
C:\Users\josia\AppData\Local\Programs\Python\Python313\Scripts
C:\Program Files\PowerShell\7\
```

- Jupyter notebooks – This is the main package required to host our interactive notebook app. To install, open your command line and simply type: “pip install jupyter.” Follow the installation process in the terminal.
- NumPy – Provides a backing library for arrays and other features used by remaining libraries. Use pip to install: “pip install numpy.”
- Pandas – Used for data processing and analyzing. Installation: “pip install pandas”
- Matplotlib – Used to create simple charts and visualizations. Installation: “pip install matplotlib”
- Seaborn – Expands on matplotlib to create detailed visualizations. Installation: “pip install seaborn”
- SciKitLearn – Provides a library to create and train our machine learning model. Installation: “pip install scikit-learn”
- IPyWidgets – Should be automatically installed with Jupyter, but you might need to run an update if widgets are not displaying: “pip install --upgrade ipywidgets jupyterlab_widgets”

Running the Program:

1. Download the application zipped archive.
2. Extract the archive to a suitable location.
3. Open the application folder and right-click the background to open folder options.
4. Select “Open in Terminal” from the right-click menu. A PowerShell window should open.
5. Type “jupyter notebook” in the terminal and press enter to open folder in jupyter.
6. Once the browser window opens the folder, select “Application.ipynb” to open the application notebook.
7. Select “Kernel/Restart Kernel” from the dropdown menu to reset the application. This can also be used to reset the application if you encounter errors.
8. Select “Run/Run All Cells” from the dropdown menu to run the application.
9. There are visualizations and explanations for every step of the data processing and model training process.
10. Scroll to the bottom of the notebook to the User Interface section.
11. Use the widgets to input demographic data and get a cost prediction.