

C964 Computer Science Capstone

09.19.2020

Mark Horn

Student ID: 001069917

Bachelor of Science, Computer Science



Table of Contents

Section A - Project Proposal - Executive	3
A.1 Letter of Transmittal	3
A.2 Problem Summary	4
A.3 Application Benefits	4
A.4 Application Description	4
A.5 Data Description	5
A.6 Objective and Hypotheses	5
A.7 Methodology	5
A.8 Funding Requirements	6
A.9 Stakeholders Impact	7
A.10 Data Precautions	7
A.11 Developer's Expertise	7
 Section B - Project Proposal - Technical	 8
B.1 Problem Statement	8
B.2 Customer Summary	8
B.3 Existing System Analysis	8
B.4 Data	9
B.5 Project Methodology	10
B.6 Project Outcomes	11
B.7 Implementation Plan	12
B.8 Evaluation Plan	12
B.9 Resources and Costs	13
B.10 Timeline and Milestones	14

Section C - Data Product - Application	15
C.1 The Application Demo	15
Section D - Post Implementation Report	20
D.1 Project Purpose	20
D.2 Datasets	20
D.3 Data Product Code	22
D.4 Hypothesis verification	24
D.5 Effective Visualizations and Reporting	24
D.6 Accuracy analysis	26
D.7 Application Testing	27
D.8 Application Files	28
D.10 User's Guide	29
D.11 Summation of Learning Experience	32
Section E - Sources	33

To: Michael Scott <mscott@globatek.com>; Executives <executives@globatek.com>

From: Mark Horn <mhorn@realdatasolutions.com>

Date: September 17th, 2020

Subject: Medical Cost Prediction Solution

Dear Michael Scott,

I was delighted to speak with you the other day and even more so to formally submit our proposal today.

As discussed, the problem Globatek is facing is the need for an in-house solution for determining potential medical costs for customers seeking insurance. Proprietary solutions have many pros and very few cons.

The proposed solution will feature a modern, attractive and flexible interface that connects with a secure data driven prediction system. Each customer will provide details such as age, height, weight and medical conditions which will be submitted by your agents. The predictions will be generated by analyzing hundreds of thousands of data records. The product will provide an estimated cost associated with each of your potential customers, in real time, which benefits both the company and the customer.

The necessary funding to complete the proposed solution is \$33,763.25. This up-front cost covers planning, development, testing and documentation. While this is costing the company up-front, there will be no ongoing fees for outsourcing or maintenance and will save Globatek money in the future.

I am very pleased that you have considered us for your solution and I will eagerly await your response. Please do not hesitate to call if you have any further questions.

Sincerely,

Mark Horn



Chief Technology Officer, Real Data Solutions

Section A

Project Proposal - Executive

Problem Summary

Globatek is in need of an in-house solution to instantly deliver a quote to potential insurance customers. The current arrangement of submitting an application and having a representative outsource cost analysis for has become dated. Most other insurance companies are delivering quotes using modern technologies and our solution will seek to establish a footprint in the modern insurance world.

Application Benefits

The proposed solution will benefit Globatek in several ways. Cutting edge technology ensures near instant response times to customer quote requests. This is significant in that it creates a positive experience for customers as well as agents. Call center engagement with customers will be significantly reduced. Furthermore, the cost benefit of owning your own solution can not be overlooked. While there is an up-front cost for the solution, Globatek will benefit financially in the long run, by not paying fees to third party sources.

Application Description

The application consists of two distinct solutions working together to achieve the business objective of providing accurate data driven insurance quotes.

The front-end solution will be built with the React platform. React offers an expressive and flexible user interface, which will act as the dashboard visible to agents.

The back-end solution will be driven by Python, which is the popular choice for data driven back-end applications. The service will handle all queries from the front-end by utilizing powerful machine learning algorithms to read historical data and make accurate predictions based on that data.

Data Description

The data that will anchor the back-end of the solution is a published dataset that contains columns for age, sex, height, weight, body mass index, children, region and health specific features such as smoking. The 'charges' field is the dependent variable, and all other fields are the independent variables.

The data is in CSV (comma separated) format, which contains fields in string, boolean, float and integer formats. All loaded data will be cleaned and converted to numerical format making it compatible with machine learning libraries.

The target data has no anomalies and zero null values. The only limitation of the data is the number of records in the set. This will correct itself over time as more and more data is added.

Objective and Hypotheses

The primary desired objective of the solution is to provide the engine with an input of variables and receive back an output that represents the amount the company expects to pay in medical costs on the customer over the course of the insurance policy.

The hypothesis is that *if* the end-user provides accurate and complete input data, *then* the system will provide accurate and complete output data within the margin of error.

Methodology

The project methodology to be used will be the Waterfall method. Waterfall methodology is tried and true. It offers a more straightforward approach with a linear style. Project timelines and deliverables are decided on up front and executed through phases which aligns well with our solution.

The phases will consist of requirements, design, implementation, verification and maintenance. In more common terms, we will plan, code, test, deploy and maintain. The business requirements are clear and will not require ongoing client input or additional features making Waterfall methodology the clear choice.

Funding Requirements

The project's total funding requirements is \$33,763.25. This cost includes the server rack and a modular Dell PowerEdge server, including installation hardware. Additionally, the personnel required includes four full time software developers over a span of four weeks and two quality control testing personnel over a span of 1 week. Planning and design are included in the total funding requirement cost. All programming tools and licensing are included in the development cost.

Please refer to the table below for a full breakdown:

Item	Unit Cost	Quantity	Duration	Cost
Rackmount 42-u 36" Glass (Server Rack)	\$896.25 one-time cost	1		\$896.25
Dell PowerEdge FC830 with Xeon E5 CPUs (Server)	\$10,859.00 one-time cost	1		\$10,859.00
Rack and Server Installation (IT Personnel)	\$25.00 per hour	2	16 hours	\$800.00
Back End Development (Developers)	\$55.00 per hour	2	160 hours	\$8,800.00
Front End Development (Developers)	\$55.00 per hour	2	160 hours	\$8,800.00
Testing (White and Black Box QA Personnel)	\$25.00 per hour	2	40 hours	\$2,000.00
Subtotal				\$32155.25
5% Contingency				\$1,607.76
Grand Total				\$33,763.25

Stakeholders Impact

Our solution will make a significant impact on each stakeholder with a vested interest, both directly and indirectly. Agents of the company will have their customer engagement minimized, while also cutting back on third-party interactions. The time saved allows agents to invest time in other crucial activities. Customer stakeholders will be able to receive their quotes with minimal time investment and a streamlined experience. The world has evolved and customers want instant gratification. Executive stakeholders will enjoy the reduction of weekly, monthly and annual costs. The solution gives the business stakeholders a significant advantage over competitors who are still on older systems, while establishing its presence among more modern competitors.

Data Precautions

Upon deployment, the datasets acquired will be a part of the Globtek data bank. As the owner of a new data driven solution, new data collected will be added daily as a result of collections from daily business.

Fortunately, the initial dataset and data acquired as a result of future business does not have legal or ethical considerations. All of the data is anonymous, and even if stolen or hacked, each record could represent a purely fictional character and not a real person. No names or personal information is stored as part of the cost prediction dataset.

Regardless of the data precautions needed, Real Data always ensures security and privacy with encrypted connections and cutting edge back-end technologies.

Developer's Expertise

Real Data Solutions has been delivering data driven solutions for our clients for over 15 years. Our track record includes delivering over 400 solutions to over 250 clients. We have a 4.9 star rating and 95% approval rating on Glassdoor.com, an online hub for tracking company reviews and ratings. The business requirements and objectives of Globatek for this project are precisely what Real Data Solutions specializes in. We employ full stack engineers, machine learning experts and a plethora of other IT professionals.

Section B

Project Proposal - Technical

Problem Statement

Globatek currently uses a call center stocked with phone agents as well as a website to collect requests for quotes. Agents in turn, submit the query to a third-party for underwriting the insurance cost analysis. Finally, the agent would respond to the customer via the original pipeline with the policy and price.

Globatek would like to move to an instantaneous quote system for agents to cut down on both cost and time. The new system needs to accept queries on a front-end system and relay them to an internal back-end machine learning algorithm for processing. The input will be submitted and compared with a large dataset of past cost analysis. The back-end needs to instantly predict and respond with an accurate representation of the quote amount.

Customer Summary

The implementation of the solution will not be public facing and will be intended for internal use only. The “end-user”, which includes company agents, employees and other personnel who will use the application are the intended audience.

The solution is a one size fits all solution and will not require a specific skill set to use. End-users will be able to access the system from any environment with a secure internet connection and a secure browser. Authorized end-users will be given credentials to access the solution.

Existing System Analysis

Globatek is currently using cloud hosting and off-site resources to conduct business. They are not utilizing their own on-site hardware or software. The call center will remain in place, but the method and means of providing insurance quotes will be replaced by our solution. Since there will be a full replacement, integration challenges will be minimal.

Data

The initial dataset is public knowledge, found on github and is in the CSV format.

<https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv>

This initial dataset only serves as the launch data of the solution. The dataset will continue to grow as new records are added. The data contains multiple columns with multiple data types. This data is not usable in its current state.

A view of the raw data:

age	sex	bmi	children	smoker	region	charges
int64	string	float	int64	string	string	float
19	female	27.9	0	yes	southwest	16884.94

Multiple libraries will be used to make this initial dataset usable. Specifically, the 'pandas', 'numpy' and scikit-learn python libraries. Ideally, all fields would be in a numerical format, either in integer or float format. Scikit-learn has functionality to convert column fields into usable machine learning data in LabelEncoder and OneHotEncoder. For purposes of our solution, we will be using LabelEncoder.

A view of the data after transformation:

X						y
age	sex	bmi	children	smoker	region	charges
int64	int64	float	int64	int64	int64	float
19	1	27.9	0	0	0	16884.94

Fortunately, we are able to skip a step here, as our dataset contains no null values, outliers or incomplete data. In a less organized dataset we would need to either discard rows with null values by dropping them, or make a best guess to fill them. Generally, if it was a low percentage of null values, the mean() of the column could be taken and used to fill those fields. Another option would be to drop the row entirely, but would impact the size of the dataset depending on how many rows would need to be dropped. It is fortunate that this is not an issue with the current dataset.

Project Methodology

This project will utilize waterfall methodology. The Globatek business requirements are clear and not likely to change. Each phase will be completed in order, with each stage beginning only on completion of the previous stage.

Requirements:

All requirements need to be gathered at this phase. The project scope and requirements documents will be drafted and submitted to Globatek for review. Once the executives sign off on the requirements document, the rest of the project can resume without further input from the client.

Design:

Our design phase is best broken up into two parts, specifically, front-end and back-end design. It is crucial that both teams communicate and support the other teams technical requirements. The front-end team will design a flexible user interface to interact with the back-end. The back-end team will design a robust and accurate machine learning API in python. Both design plans should be submitted to project management for verification of business requirements and sign-off.

Implementation:

The program developers will assimilate the specifications and requirements from the requirements and design phases and begin implementation of those designs. This stage has a time allotment of 20 business days or 4 weeks. In this stage, the rack server as well as the Dell PowerEdge server will be installed by the IT department. The front and back end code bases will be installed on the new server infrastructure.

Verification:

In this stage the QA department will begin the testing of the implementation of the solution. One QA team member will focus solely on white box testing, and rigorously test the code, internal functions and unit testing. One QA team member will be dedicated solely to black box testing and focus on functionality only, from an end-user perspective.

QA team members will communicate through bug trackers and direct communication with the developers. After adequate testing by the QA team, begin rolling out beta testing to Globatek employees in a limited capacity.

Maintenance:

The client should be regularly using the product during this maintenance phase, and communicating any discovered bugs, inadequate features and any other errors found. The developers will address these fixes as needed until the customer is satisfied. This is the acceptance testing and will signify the end of the project.

Project Outcomes

Project outcomes is an itemized list of all expected deliverables, including both project and product deliverables as explained below.

Project Deliverables:

Business requirements document.

Project Scope - Timeline, product deliverables.

Kickoff meeting - Sign off documents officially launching the project.

Back-end design plan - Sample Python code, mockups and illustrations.

Front-end design plan - Sample React javascript code, mockups and illustrations.

Testing plan and bug tracker - include all bugs found and results.

Product Deliverables:

Back-end application, powered by Python.

Front-end application, powered by React (JS).

Machine learning models constructed during development.

All supporting product documentation.

** All product deliverables include full source code.

Implementation Plan

The general strategy for moving the data product into the production environment consists of multiple parts.

The first phase of the rollout will begin in the verification stage of the methodology upon completion of the QA testing. The product at this stage has undergone rigorous testing in a test environment, but not a live environment. After the engineers have installed both services in the newly installed PowerEdge server, the product should be made available to 25% of the insurance agents at Globatek. The old system is still in place as a fail safe.

Developers should work closely with the bug tracker as well as communicate directly with the end-users where necessary, and work to resolve issues as quickly as possible.

After 72 hours of no reported bugs and a full success rate on prior bugs and feature requests, the number of agents should be increased to 50%. Repeat phase 1 of the beta rollout. Finally, the full implementation.

Levels of testing for beta users including testing credentials and the login system, navigating the menus, and submitting quotes. Surveys will be emailed to each user at the end of the subsequent addition of users that will provide feedback on functionality, performance and overall satisfaction of the product in the live beta environment.

The final milestone of the verification phase is when 100% of the end-user base is completely using the new system and not the old system and zero bugs and complaints have been reported over a 72 hour time span.

Evaluation Plan

Much of the evaluation plan has already been completed in the implementation phase, due to the nature of the rollout of beta testing. A meeting will be held between project management and the executive team of Globatek to review the business requirements documentation, and the survey results from the end-users. At this stage there should already be 100% positive feedback with no bugs and no additional feature requests. This meeting should be just a formality to demonstrate to the executive team that all requirements have been met, and all end-users are satisfied with the product.

Resources and Costs

The project's total funding requirements is \$33,763.25. This cost includes the server rack and a modular Dell PowerEdge server, including installation hardware. Additionally, the personnel required includes four full time software developers over a span of four weeks and two quality control testing personnel over a span of 1 week. Planning and design are included in the total funding requirement cost. All programming tools and licensing are included in the development cost.

Please refer to the table below for a full breakdown:

Item	Unit Cost	Quantity	Duration	Cost
Rackmount 42-u 36" Glass (Server Rack)	\$896.25 one-time cost	1		\$896.25
Dell PowerEdge FC830 with Xeon E5 CPUs (Server)	\$10,859.00 one-time cost	1		\$10,859.00
Rack and Server Installation (IT Personnel)	\$25.00 per hour	2	16 hours	\$800.00
Back End Development (Developers)	\$55.00 per hour	2	160 hours	\$8,800.00
Front End Development (Developers)	\$55.00 per hour	2	160 hours	\$8,800.00
Testing (White and Black Box QA Personnel)	\$25.00 per hour	2	40 hours	\$2,000.00
Subtotal				\$32155.25
5% Contingency				\$1,607.76
Grand Total				\$33,763.25

Timeline and Milestones

Phase: Milestone	Duration	Start	End	Resources
Requirements: Project Scope	1 day (8 hours)	10/14/2020	10/14/2020	Exec, PM, Dev, QA, IT
Planning: Project Plan	2 days (16 hours)	10/15/2020	10/16/2020	PM, Dev, QA
Planning: Kickoff meeting	1 day (4 hours)	10/19/2020	10/19/2020	Exec, PM, Dev, QA, IT
Design: Back-end Design Plan	2 days (16 hours)	10/20/2020	10/21/2020	PM, Dev
Design: Front-end Design Plan	2 days (16 hours)	10/22/2020	10/23/2020	PM, Dev
Design: Testing Plan	1 day (8 hours)	10/23/2020	10/23/2020	PM, QA
Implementation: Server Installation	1 day (8 hours)	10/26/2020	10/26/2020	IT
Implementation: Back-end Development	30 days (160 hours)	10/26/2020	11/23/2020	Dev, IT
Implementation: Front-end Development	30 days (160 hours)	10/26/2020	11/23/2020	Dev, IT
Verification: QA Testing	5 days (40 hours)	11/27/2020	12/4/2020	QA
Verification: Beta Testing	5 days (40 hours)	12/7/2020	12/11/2020	PM, Dev, EU
Verification: Deployment	2 days (16 hours)	12/14/2020	12/15/2020	PM, Dev, IT
Verification: Executive signoff	1 day (4 hours)	12/15/2020	12/15/2020	Exec, PM
Maintenance: Documentation & Training	1 day (8 hours)	12/16/2020	12/16/2020	Exec, PM, Dev, QA, IT

Section C

Data Product - Application

Demo:

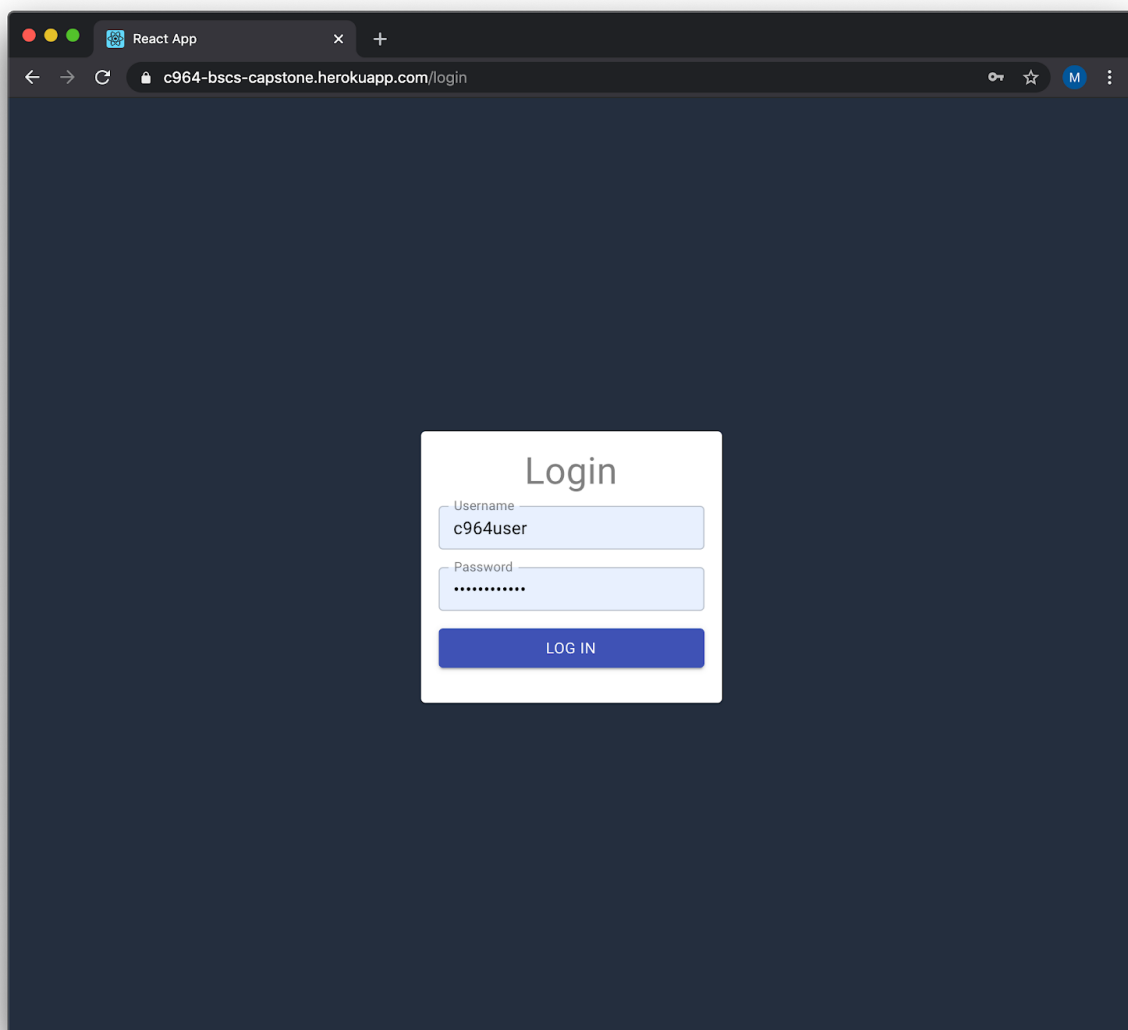
URL: <https://c964-bscs-capstone.herokuapp.com/app>

Credentials:

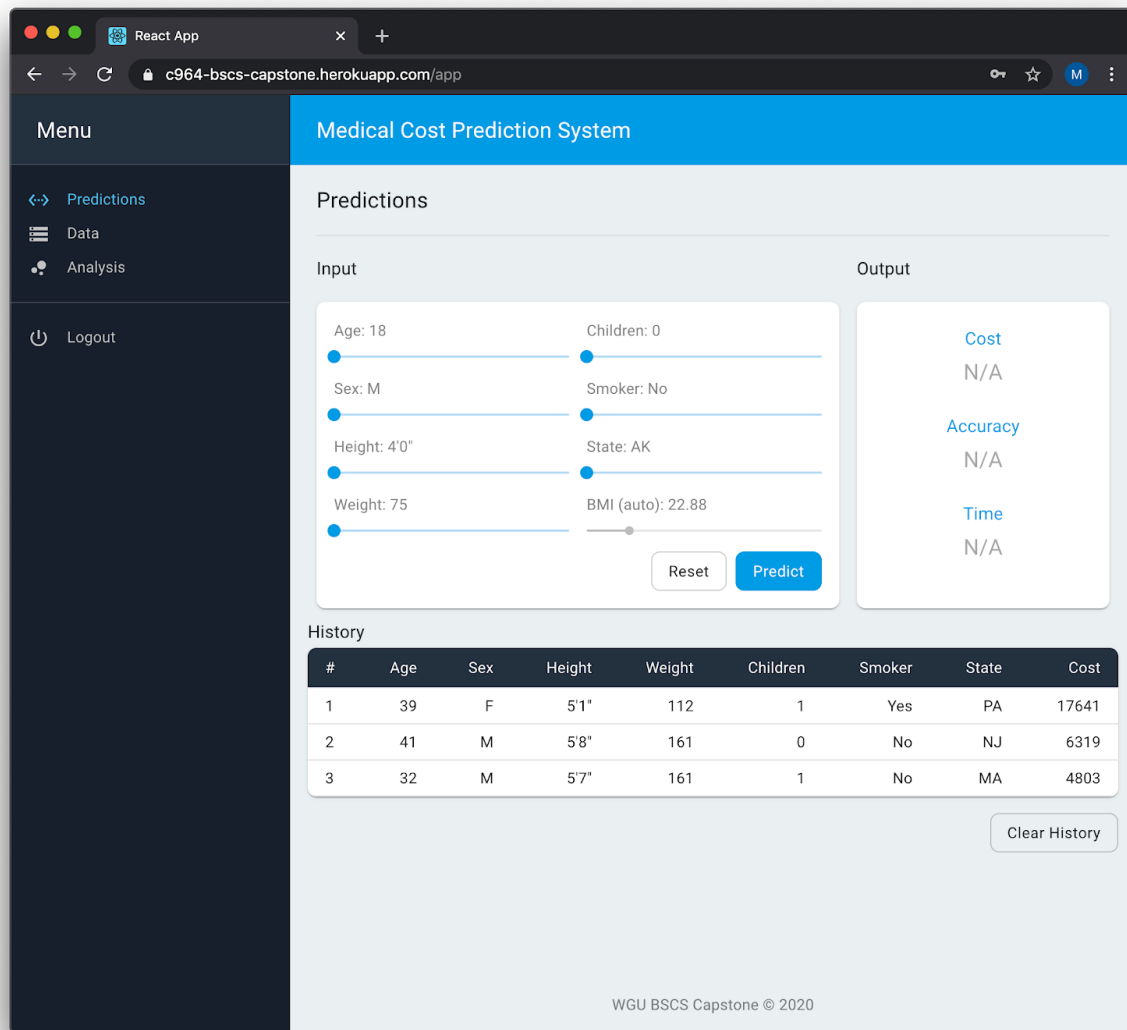
Username: c964user

Password: c964password

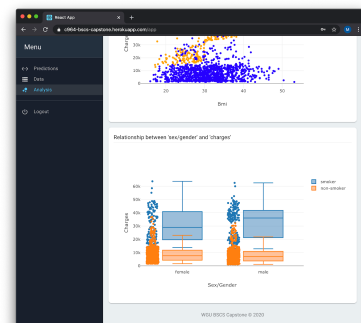
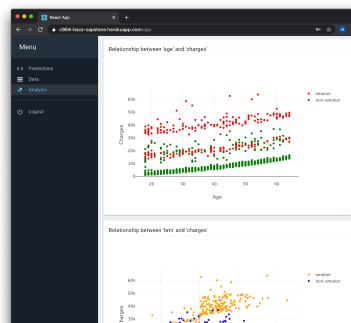
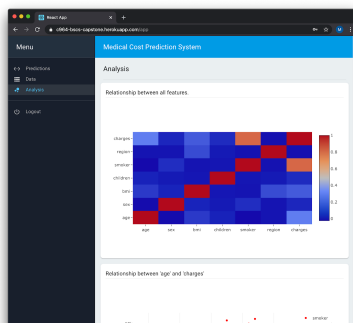
Login Page:



The Dashboard:



Analysis Page:



Data Page:

The screenshot shows a web application titled "Medical Cost Prediction System". The interface has a dark sidebar menu on the left with options: "Menu", "Predictions", "Data" (selected), "Analysis", and "Logout". The main content area displays a table of data with 13 rows and 7 columns: age, sex, bmi, children, smoker, region, and charges. Below the table, there are two sections: "Column Descriptions" and "Notes".

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.924
18	male	33.77	1	no	southeast	1725.5523
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47061
32	male	28.88	0	no	northwest	3866.8552
31	female	25.74	0	no	southeast	3756.6216
46	female	33.44	1	no	southeast	8240.5896
37	female	27.74	3	no	northwest	7281.5056
37	male	29.83	2	no	northeast	6406.4107
60	female	25.84	0	no	northwest	28923.13692

Rows per page: 10 1-10 of 1338

Column Descriptions

- age:** Age of primary beneficiary
- sex:** Male or female
- bmi:** Body mass index, (kg/m²) or (w/h²)*703
- children:** Number of children/dependents
- smoker:** Is the beneficiary a smoker or non-smoker
- region:** Residential area in the US
- charges:** Medical costs billed by health insurance

Notes

- The dataset displayed is in raw form. It has not been cleaned and is not representative of the final data the prediction model sets are based on.
- The dataset can be found at <https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv>

Description:

The front-end was developed in React, an open-source javascript framework. The user interface is a mostly custom interface from scratch, but utilizes some open-source Material UI components.

The user interface is an SPA (single page application) and utilizes a router to redirect all requests to the domain to /app. The router creates the content on a per page basis and displays it in the /app wrapper.

Sample React Code:

```

JS index.js  ×
src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5  import * as serviceWorker from './serviceWorker';
6
7  ReactDOM.render(
8    <App />,
9    document.getElementById('root')
10 );
11
12 serviceWorker.register();
13

```

React is a stateful framework. Here is a snippet from the navigator code. Each component in the single page application accepts properties and states from the parent component allowing it to pass state between components. The css or stylesheet of the application can be written inline, declared before each component or passed statefully as props as seen in the image below:

```

JS Navigator.js  ×
src > JS Navigator.js > ...
73
74 function Navigator(props) {
75   const { classes, ...other } = props;
76   let [active, setActive] = useState(0);
77
78   let change = (index) => {
79     setActive(index);
80     props.navigate(index);
81   }
82
83   return (
84     <Drawer variant="permanent" {...other}>
85       <AppBar
86         component="div"
87         className={classes.categoryHeader}
88         color="primary"
89         position="static"
90         elevation={0}>
91         <Toolbar>
92           <Typography color="inherit" variant="h6" nowrap="true">
93             Menu
94           </Typography>
95         </Toolbar>
96       </AppBar>

```

The Router:

This code eliminates the need for error pages such as 401 and 404 HTML error codes.

The router will redirect to an authorized path, while honoring the authentication.

```

29   return (
30     <Router>
31       <Switch>
32         <Route path="/login">
33           {authenticated ?
34             <Redirect to={{pathname: "/app"}}/>
35             :
36             <Auth login={() => login()}/>
37           }
38         </Route>
39
40         <Route path="/app">
41           {authenticated ?
42             <Main logout={() => logout()}/>
43             :
44             <Redirect to={{pathname: "/login"}}/>
45           }
46         </Route>
47
48         <Route path="/*">
49           {authenticated ?
50             <Redirect to={{pathname: "/app"}}/>
51             :
52             <Redirect to={{pathname: "/login"}}/>
53           }
54         </Route>
55       </Switch>
56     </Router>
57   );
58 }
59
60

```

Content is rendered on-screen according to the index of the router props:

```

31   function Content(props) {
32     const { classes } = props;
33
34     return (
35       <div>
36         {props.index === 0 &&
37           <div><PredictionPage/></div>
38         }
39         {props.index === 1 &&
40           <div><DataPage/></div>
41         }
42         {props.index === 2 &&
43           <div><AnalysisPage/></div>
44         }
45       </div>
46     );
47   }
48
49   Content.propTypes = {
50     classes: PropTypes.object.isRequired,
51   };
52
53   export default withStyles(styles)(Content);

```

Section D

Post Implementation Report

Project Purpose

The problem as identified by Globatek was they wanted a modernized prediction system that could assist agents in providing instant accurate quotes to potential insurance customers. In addition, they wanted to eliminate the need for 3rd party outsourcing of their cost analysis. Globatek's previous system was no longer meeting the company's vision and business requirements.

The objective of the product was to provide an intuitive, scalable and flexible application to assist agents with providing potential customers with accurate quotes provided by the machine learning algorithms on the back-end. We succeeded in meeting all of Globatek's expectations. We provided them with a front-end dashboard SPA (single page application) that served as a hub for providing quotes for customers, as well as visualizations of current and previous data. The dashboard also provided data visualizations of the Random Forest Regression algorithm on the stored data.

Datasets

The raw dataset is a historical dataset in CSV (comma separated) from the following:
<https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv>

A view of the raw csv file:

```
age,sex,bmi,children,smoker,region,charges
19,female,27.9,0,yes,southwest,16884.94
```

A view of the raw csv file in a table:

age	sex	bmi	children	smoker	region	charges
int64	string	float	int64	string	string	float
19	female	27.9	0	yes	southwest	16884.94

Next, we need to check if there are null values and convert categorical features to numbers. We need to check for null values, and decide what to do, if applicable:

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1338 non-null   int64
1   sex         1338 non-null   object
2   bmi         1338 non-null   float64
3   children    1338 non-null   int64
4   smoker      1338 non-null   object
5   region      1338 non-null   object
6   charges     1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

Fortunately, no null values in our dataset, so we can move right into conversions:

```
from sklearn.preprocessing import LabelEncoder

data[['sex', 'smoker', 'region']] =
data[['sex', 'smoker', 'region']].astype('category')

label = LabelEncoder()

label.fit(data.sex.drop_duplicates())
data.sex = label.transform(data.sex)

label.fit(data.smoker.drop_duplicates())
data.smoker = label.transform(data.smoker)

label.fit(data.region.drop_duplicates())
data.region = label.transform(data.region)

data.dtypes

age           int64
sex           int64
bmi          float64
children     int64
smoker       int64
region       int64
charges     float64
dtype: object
```

All data is now in numerical and categorized format compatible with machine learning.

A view of the data after transformation:

X						y
age	sex	bmi	children	smoker	region	charges
int64	int64	float64	int64	int64	int64	float64
19	1	27.9	0	0	0	16884.94

Data Product Code

The functionality of the code used to analyze the data and provide the necessary descriptive and predictive outputs as follows: We opted to implement both Linear Regression and Random Forest regression. The entire code base is included in the submission, and some examples of the code base are included below:

First, we need to split the data into train and test sets and verify the shape:

```
from sklearn.model_selection import train_test_split

X = data.drop(["charges"], axis=1)
y = data["charges"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2)

X_train.shape, X_test.shape, y_train.shape, y_test.shape

((1070, 6), (268, 6), (1070,), (268,))
```

Second, establish a baseline using the Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=5)
model.fit(X_train, y_train)
model.score(X_test, y_test)

0.811155752579035
```

An accuracy of 81% is not bad, but could potentially be improved with some tuning

Next, we will tune the estimators to try and improve the accuracy:

```

from sklearn.metrics import r2_score, mean_squared_error

np.random.seed(99)
for i in range(10, 100, 10):
    model = RandomForestRegressor(n_estimators=i).fit(
        X_train, y_train)
    print(f"Accuracy with {i} estimators: {
        model.score(X_test, y_test) * 100}%")

```

```

Accuracy with 10 estimators: 83.73902631735224%
Accuracy with 20 estimators: 83.68222858658231%
Accuracy with 30 estimators: 85.10173832461967%
Accuracy with 40 estimators: 86.75884841510994%
Accuracy with 50 estimators: 86.08489272200673%
Accuracy with 60 estimators: 89.14801255231139%
Accuracy with 70 estimators: 87.1942867036696%
Accuracy with 80 estimators: 86.38958626038517%
Accuracy with 90 estimators: 85.96458201939544%

```

Using 60 estimators in the random forest clearly the best result for our dataset.

```
r2_score(y_test, model.predict(X_test))
```

```
0.8880495819912851
```

Now, we are able to save and load tuned model:

```

import pickle
pickle.dump(model, open("model.pkl", "wb"))
model = pickle.load(open("model.pkl", "rb"))

```

```
RandomForestRegressor()
```

Finally, we can now input variables into the model for a result (y=charges):

```

# age, sex, bmi, children, smoker, region
# 28 year old male, bmi of 25, with no children and doesn't smoke
model.predict([[28,0,25,0,0,0]])

```

```
array([3881.61])
```


Hypothesis verification

The primary desired objective of the solution is to provide the engine with an input of variables and receive back an output that represents the amount the company expects to pay in medical costs on the customer over the course of the insurance policy.

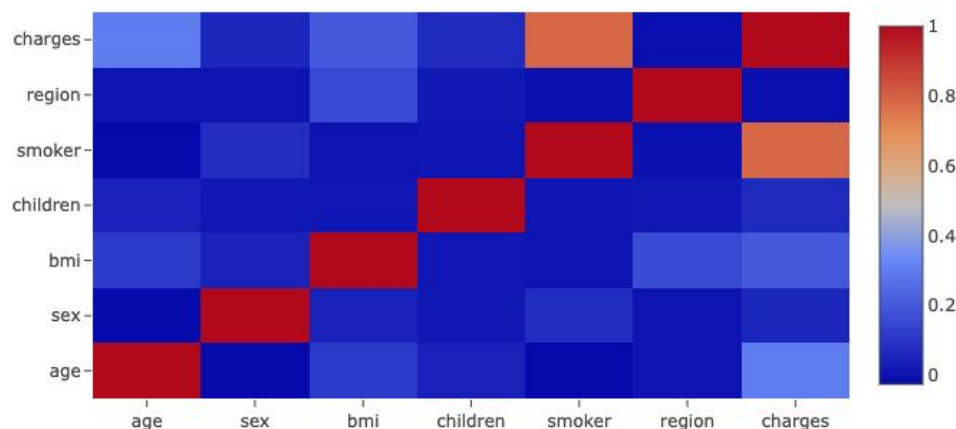
The hypothesis was that *if* the end-user provides accurate and complete input data, *then* the system will provide accurate and complete output data within the margin of error.

We wanted to provide predictions with a minimum accuracy of 80%, which coincidentally was surpassed even in the baseline model. Tuning resulted in achieving approximately 89%. The assumptions about the phenomenon in fact proved true.

Effective Visualizations and Reporting

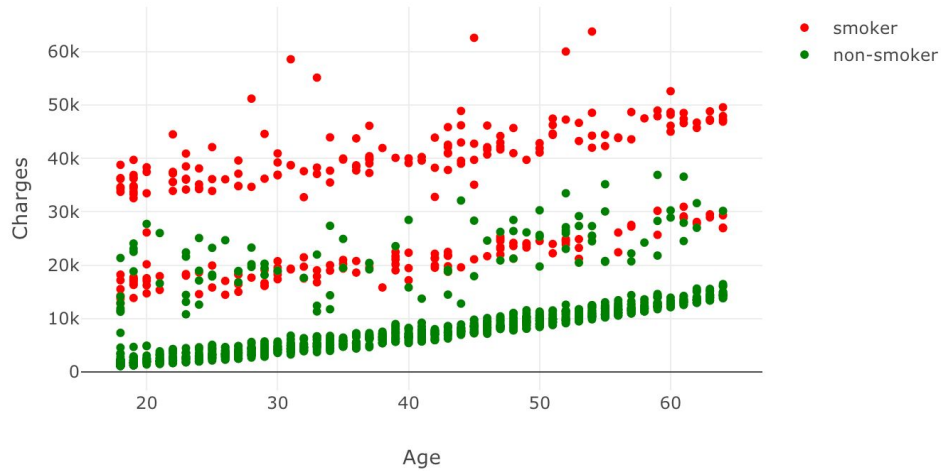
We included a full page dedicated to data visualization which we feel touched on the most important features to our dataset and painted a clear picture of the story the data was telling.

First, we wanted to plot correlation to see which X input variables most correlated with the expected y output. The plot suggests that smoking most closely relates to charges by a large margin. It was surprising to see that age took the second spot over bmi and children.



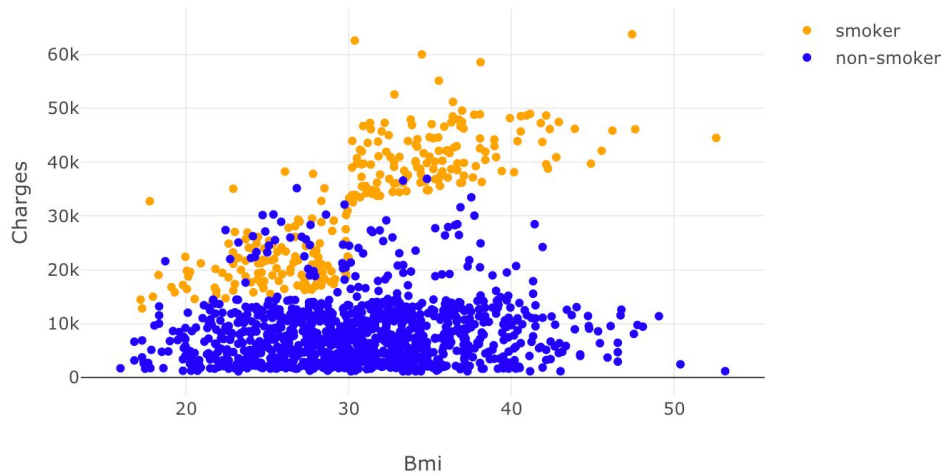
Since smoking was very strong, we decided to plot it each of the subsequent plots as a sub-variable and analyze:

Relationship between 'age' and 'charges'



Body mass index (BMI) was a bit surprising as it did not correlate with charges as strongly as smoking and age did.

Relationship between 'bmi' and 'charges'



Lastly, we decided to analyze if medical charges were related to sex or gender in a meaningful way. They aren't.

Relationship between 'sex/gender' and 'charges'



Accuracy analysis

To provide the most precise accuracy we tested it multiple ways including the scikit-learn built in scoring method, cross validation and r2 scoring:

Model	.score	.cross_val_score	.r2_score
LinearRegression	.89	.87	.88
RandomForestReg	.88	.88	.91

Each of these methods of scoring provided a high enough accuracy to validate the hypothesis target of 80% accuracy .

Application Testing

Testing for our product was broken into several phases. We implemented a QA test plan, as well as a beta test plan for end-users.

1. QA Testing

Testing in this phase (Implementation) consisted of about 40 hours of testing and 2 unique QA testers, in addition to the code developers. The participants were solely focused on unit and system testing.

One QA tester was assigned white box testing and focused on exclusively unit and system testing.

The second QA tester was assigned exclusively black box testing, and focussed on pure functionality from an end-user perspective.

2. Beta Testing

The second phase of testing (Verification Phase) was rolled out incrementally to a small percentage of actual employees at Globatek. A bug tracking log and feature request log were kept. The participants were focussed on acceptance testing.

The number of users in the beta test were incremented after 72 hours of no bugs or reasonable requests were made.

The beta ended when 100% of the current Globatek agents were using the system and the bug tracker was completely cleared.

Application Files

Backend

/root

app.py
model.pkl
requirements.txt
Procfile

/data

Insurance.csv

root directory

public facing API
machine learning model
dependencies
server configuration file

data directory

comma separated data file

FrontEnd

/root

package.json

/public

/node_modules

/src

App.js
App.test.js
Auth.js
Content.js
Header.js
Index.js
Main.js
Navigator.js
serviceWorker.js
States.js

/Data

/Pages

Analysis.js
Data.js
Predictions.js

root directory

react configuration file

public facing html directory

node environment directory

react source directory

single page application wrapper
internal react tests
user authorization and login
wrapper for /Pages
the header module
root application file
wrapper for Content.js
Router navigation code
browser service worker
state and region data

data directory

pages directory

page for displaying the data analysis and plots
page for displaying the raw data
page for displaying the prediction controls

User's Guide

There is no installation required for the application because it is a web app that was pre-installed by the IT department. Usage guide below:

Important Note: This application demo is hosted on heroku 'free version'. Because of this the servers are put into an idle state when not in use.

- **Upon visiting the site, it will respond slowly while the react server boots.**
- **Upon 1st query, the server will respond slowly, while the python server boots.**
- **Subsequent logins and queries will perform at normal speed.**

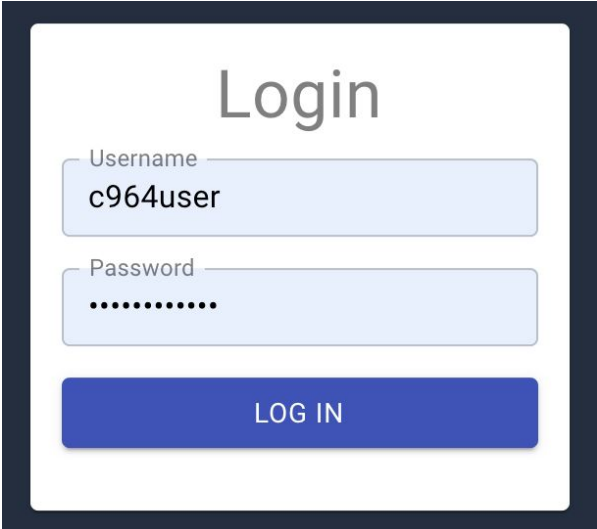
To begin, navigate to: <https://c964-bscs-capstone.herokuapp.com/>

Login component:

First, the application is protected by an authentication system, and you must enter credentials to gain access.

Enter the credentials Globatek has provided, or use the demo credential below:

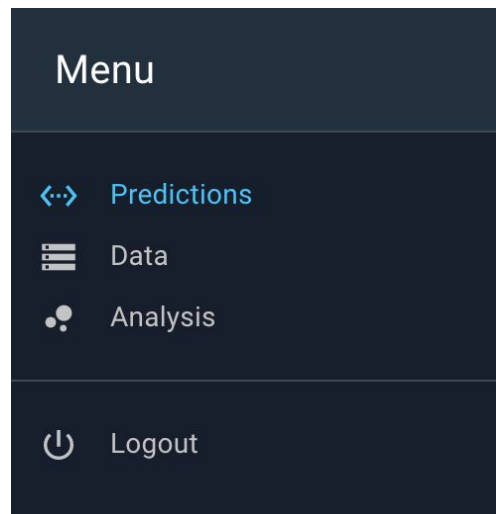
Username: **c964user**
Password: **c964password**

A screenshot of a web application's login component. It features a dark blue border around a white background. At the top, the word "Login" is displayed in a large, grey, sans-serif font. Below it, there are two input fields. The first field is labeled "Username" in a small, grey font and contains the text "c964user". The second field is labeled "Password" in a small, grey font and contains a series of black dots. Below these fields is a solid blue button with the text "LOG IN" in white, uppercase letters.

Navigation component:

The application is a single page application. All attempted redirects will route back to /app.

- Predictions: This is the input page for entering variables and receiving results
- Data: This page is for purely for viewing the raw dataset
- Analysis: This page is for viewing the data visualizations



Input component

To use the input component, simply drag the sliders to the desired positions. The reset button sets all sliders back to default values, and the predict button submits the query to the backend and awaits the result.




Input

Age: 18	Children: 0
Sex: M	Smoker: No
Height: 4'0"	State: AK
Weight: 75	BMI (auto): 22.88

Reset Predict

The Output component:

This component displays the results received back from the API.

Output	Output	Output
<div>Cost</div> <div>N/A</div> <div>Accuracy</div> <div>N/A</div> <div>Time</div> <div>N/A</div>		<div>Cost</div> <div>\$1312.00</div> <div>Accuracy</div> <div>91.69%</div> <div>Time</div> <div>0.591</div>

History component:

This component stores your past queries in the database. This history is accessible across multiple logins or sessions and will remain there until they are cleared using the Clear History button.

History								
#	Age	Sex	Height	Weight	Children	Smoker	State	Cost
1	39	F	5'1"	112	1	Yes	PA	17641
2	41	M	5'8"	161	0	No	NJ	6319

Clear History

Remaining components:

The remaining components are simply visual representations of the data tables and plots representing the machine learning and analysis. To logout, simply click the Logout button.

Summation of Learning Experience

Previous experience played a part in being able to piece together a full solution for this project. I have probably experimented in almost every programming language at least once in my career. I chose React because I had a strength in node and javascript as well as html. I had previous experience in databases, data structures and general project management which also played a key role. Because the backend was mostly machine learning I knew it was going to python due to most mainstream machine learning libraries supporting python.

I had only dabbled in reading up about machine learning in the past, so I knew going in that it was not going to be my strong suit. I never had actually coded anything machine learning related before and was not even aware of what Jupyter Notebooks was. I opted to enroll in a 50 hour machine learning course and skipped over the general education (python) sections. I familiarized myself with pandas, numpy, scikit-learn and matplotlib. Upon completion of the course I was confident I could complete the whole project.

I did not seek out help from any other individuals for aid, however I did visit the CompSci WGU subreddit, to get a grasp about how others approached the course.

The experience has contributed to my concept of life-long learning mostly on a project management scale. I have not completed a project of this scale before and covered all aspects of the solution solo. I had worked on large projects before, but always in teams and with a subset of tasks.

Overall I would conclude that I am happy with how the project turned out, and even a little bit proud of it. I look forward to using the new skills I acquired during this project in the future.

Section E

Sources

No external sources cited