Student Performance Analysis

FINAL PROJECT REPORT

JASONMA
160 pleasant st.
Malden, MA and 02148

VERSION 0.1.0

08/23/2020

VERSION HISTORY

VERSION	APPROVED BY	REVISION DATE	DESCRIPTION OF CHANGE	AUTHOR
0.1	Jason MA	Aug.10.2020	Initial	Jason MA
0.2	Jason MA	Aug.13.2020	Preprocess data	Jason MA
0.3	Jason MA	Aug.17.2020	Exp1	Jason MA
0.4	Jason MA	Aug.20.2020	Exp2	Jason MA
1.0	Jason MA	Aug.23.2020	Finish	Jason MA

PREPARED BY	Jason Ma	TITLE	Author	DATE	Aug.23.2020
APPROVED BY	Jason Ma	TITLE	Author	DATE	Aug.23.2020

ROLES AND RESPONSIBILITIES

NAME	ROLE	RESPONSIBILITIES
Jason MA	Author	Exploring data, cleansing data, process data, design network, train model, test model, documentation.

Project Summary

In this project, I dive into a student grades dataset from a Portuguese high school. I tried to figure out what behaviors or features impact students' grades most. Would it be the same with many parents' opinions that exists some "bad" activities cause a decrease in grades. In the beginning, I pick a little data and assume a null-hypothesis and alternative-hypothesis set. By modeling the data and using a single-tailed Binomial Distribution examination, I get the conclusion with data. Then, I replace the dataset with all background information to find out the top 10 important behaviors or features in the dataset. Inspiring by the result of permutation feature importance, I decided to add past grades into the dataset and try to increase the difficulty. From Pass/ Fail binary classification, to 5 levels multi-label classification, actual score prediction in the end.

Detail of Dataset

The dataset used in this project comes from UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Student+Performance#). There are 33 columns in the original dataset. After cleansing the data, I keep 29 columns for exploring and training.

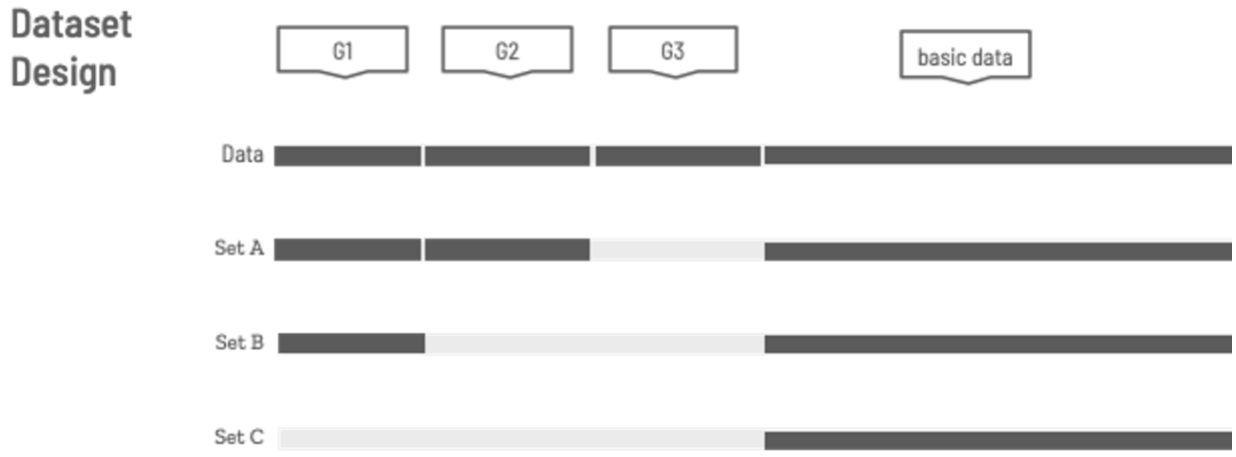
```
1 sex - student's sex (binary: "F" - female or "M" - male)
2 age - student's age (numeric: from 15 to 22)
3 address - student's home address type (binary: "U" - urban or "R" - rural)
4 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
5 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
6 Medu - mother's education (numeric: 0 - none,  1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
7 Fedu - father's education (numeric: 0 - none,  1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
8 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
9 Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
10 reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
11 guardian - student's guardian (nominal: "mother", "father" or "other")
12 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
13 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
14 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
15 schoolsup - extra educational support (binary: yes or no)
16 famsup - family educational support (binary: yes or no)
17 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
18 activities - extra-curricular activities (binary: yes or no)
19 higher - wants to take higher education (binary: yes or no)
20 internet - Internet access at home (binary: yes or no)
21 romantic - with a romantic relationship (binary: yes or no)
22 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
23 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
24 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
25 health - current health status (numeric: from 1 - very bad to 5 - very good)
26 absences - number of school absences (numeric: from 0 to 93)

# these grades are related with the course subject, Math or Portuguese:
27 G1 - first grade (numeric: from 0 to 20)
28 G2 - second grade (numeric: from 0 to 20)
29 G3 - final grade (numeric: from 0 to 20, output target)

Additional note: there are several (382) students that belong to both datasets .
These students can be identified by searching for identical attributes
that characterize each student, as shown in the annexed R file.
```

Because Portuguese high schools choose Trimester rather than Semester applied in the US, I just simply apply Trimester to a Semester pattern.

- G1 -> the final exam of the first semester
- G2 -> the midterm of the second semester
- G3 -> the final exam of the second semester (the Goal!)

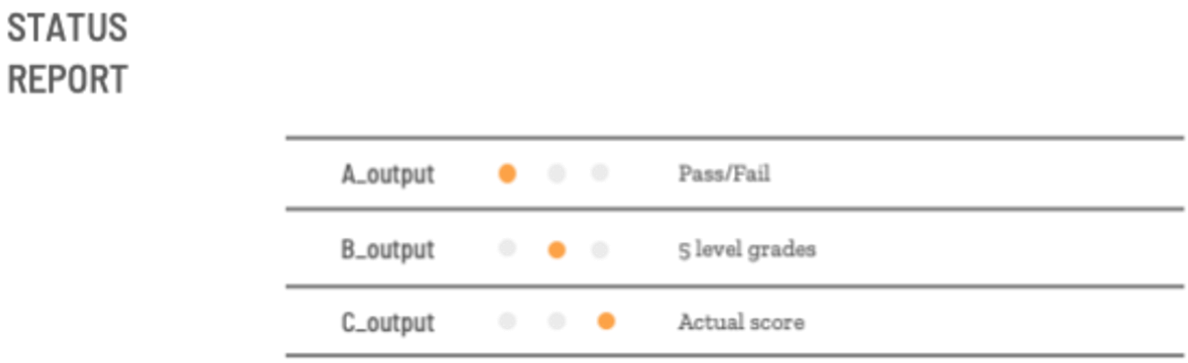
Training and testing datasets

In this project, I separate the entire dataset into three different sets. These three datasets contain different data levels.



- Set A -> basic data + G1 + G2 (Contains most of the original dataset)
- Set B -> basic data + G1 (Contains less information than Set A)
- Set C -> basic data ONLY (Contains the least information)

Similar to the data X, I also choose three different levels of the data y.



- A_output -> Pass/ Fail (The easiest one)
- B_output -> 5 levels grade (Increasing the difficulty)
- C_output -> Actual score (The most difficult one. Predicting the score in the final exam 'G3')

Workflow

1. Exploring data

Take a first look of the dataset.

```
In [128]: students_data = pd.read_csv('data/student-mat.csv', sep = ';')
          students_data.head()
```

Out[128]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	free
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	

5 rows × 33 columns

Check the type in the dataset

```
In [129]: students_data.dtypes, len(students_data)
```

```
Out[129]: (school        object
           sex           object
           age            int64
           address       object
           famsize       object
           Pstatus       object
           Medu           int64
           Fedu           int64
           Mjob          object
           Fjob          object
           reason        object
           guardian      object
           traveltime     int64
```

2. Cleansing data

2. Clean data

```
In [130]: # These features are for schools' identity. None of them will be use in this project.
          students_data2 = students_data.drop(['school', 'Dalc', 'Walc', 'nursery'], axis = 1)
          students_data2.head()
```

Out[130]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	internet	romantic	famrel	freet
0	F	18	U	GT3	A	4	4	at_home	teacher	course	...	no	no	4	
1	F	17	U	GT3	T	1	1	at_home	other	course	...	yes	no	5	
2	F	15	U	LE3	T	1	1	at_home	other	other	...	yes	no	4	
3	F	15	U	GT3	T	4	2	health	services	home	...	yes	yes	3	
4	F	16	U	GT3	T	3	3	other	other	home	...	no	no	4	

5 rows × 29 columns

3. Preprocessing data

Since we cannot use string type in modeling and one-hot encoding, replace them with integer.

```
In [131]: # check how many kinds of object type in dataset
          object_list  = ['address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian',
                          'schoolsup', 'famsup', 'paid', 'activities', 'higher', 'internet',
                          'romantic']
          print([set(students_data2[col]) for col in students_data2[object_list]])
```

```
[{'U', 'R'}, {'GT3', 'LE3'}, {'T', 'A'}, {'health', 'teacher', 'services', 'other', 'at_home
'}, {'health', 'teacher', 'services', 'other', 'at_home'}, {'other', 'course', 'reputation',
'home'}, {'other', 'father', 'mother'}, {'no', 'yes'}, {'no', 'yes'}, {'no', 'yes'}, {'no',
'yes'}, {'no', 'yes'}, {'no', 'yes'}, {'no', 'yes'}]
```

4. Creating different datasets & One-hot encoder

SetA, SetB ,SetC:

```
In [162]: setA = students_data2.drop(['G3'], axis = 1)
          setA.head()
```

```
In [163]: setB = students_data2.drop(['G2', 'G3'], axis = 1)
          setB.head()
```

```
In [164]: setC = students_data2.drop(['G1', 'G2', 'G3'], axis = 1)
          setC.head()
```

outputA, outputB, output:

Out[177]:

	fail	pass	origin
0	1	0	6
1	1	0	6
2	0	1	10
3	0	1	15
4	0	1	10
5	0	1	15
6	0	1	11
7	1	0	6
8	0	1	19
9	0	1	15

Out[182]:

	0~9	10~11	12~13	14~15	16~20	origin
0	1	0	0	0	0	6
1	1	0	0	0	0	6
2	0	1	0	0	0	10
3	0	0	0	1	0	15
4	0	1	0	0	0	10
5	0	0	0	1	0	15
6	0	1	0	0	0	11
7	1	0	0	0	0	6
8	0	0	0	0	1	19
9	0	0	0	1	0	15

Out[150]:

	0	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	origin
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	10
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	15
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	10
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	15
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	11
7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	19
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	15

5. Save to local

```
In [186]: data_dir = 'data'
          setA.to_csv(os.path.join(data_dir, 'setA.csv'), index=False)
          setB.to_csv(os.path.join(data_dir, 'setB.csv'), index=False)
          setC.to_csv(os.path.join(data_dir, 'setC.csv'), index=False)

          pd.DataFrame(outputA).to_csv(os.path.join(data_dir, 'outputA.csv'), header=False, index=False
          pd.DataFrame(outputB).to_csv(os.path.join(data_dir, 'outputB.csv'), header=False, index=False
          pd.DataFrame(outputC).to_csv(os.path.join(data_dir, 'outputC.csv'), header=False, index=False
```

6.

Experiment 1: Would my mom's concerns become true, such as puppy love and the internet will push me to fail the course? `"Bad" Activities + Pass/Fail`

My mom always cares about these bad activities:

- Activities -> extra-curricular activities (binary: yes or no)
- Internet -> internet access at home (binary: yes or no)
- Romantic -> with a romantic relationship (binary: yes or no)
- Freetime -> free time after school (numeric: from 1 - very low to 5 - very high)
- Goout -> going out with friends (numeric: from 1 - very low to 5 - very high)

Choose data y and data x from local files, then combined them

```
In [281]: bad_activities = pd.DataFrame(setC, columns=['activities', 'internet', 'romantic', 'freetime'
          bad_activities.head()
```

Out[281]:

	activities	internet	romantic	freetime	goout
0	0	0	0	3	4
1	0	1	0	3	3
2	0	1	0	3	2
3	1	1	1	2	2
4	0	0	0	3	2

```
In [375]: # split data into training and test datasets
          X_train, X_test, y_train, y_test = None, None, None, None # make sure X and y are empty
          X_train, X_test, y_train, y_test = train_test_split(bad_activities, outputA, test_size = 0.2,

In [376]: len(X_train), len(y_train), len(X_train.columns), len(y_train.columns)
Out[376]: (316, 316, 5, 2)

In [377]: len(X_test), len(y_test), len(X_test.columns), len(y_test.columns)
Out[377]: (79, 79, 5, 2)
```

7. Set null hypothesis & alternative hypothesis

Null Hypothesis:

`Predict Pass/Fail with more than 80% accuracy by using those "bad" activities.`

Alternative Hypothesis

`Prediction accuracy lower than 80%.`

Since we want to predict the result to Pass/Fail pattern, it is a classic Binomial Distribution problem. Therefore, I would use `Binomial Distribution formula` and `single-tail test` to examine the null hypothesis.

$$P(x) = C_n^x p^x (1-p)^{n-x} = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

`Significance Value: 5%`

Using Binomial Distribution Formula to find the critical area

In [379]:
```
def Binomial_Distribution(n, p):
    out = []
    critical_region = 0
    critical_value = 0.0
    for x in range(0, n + 1):
        cx = math.factorial(float(n)) / (math.factorial(float(x)) * math.factorial(float(n -
        ux = (p ** float(x)) * ((1.0 - p) ** float(n - x))
        px = round(cx * ux, 5)
        out.append(px)
        if critical_value < 0.05:
            critical_value += px
            critical_region += 1
    return out, critical_region
```

In [380]:
```
binomial, critical_region = Binomial_Distribution(len(y_test), 0.8)
binomial[:10], len(binomial)
```

Out[380]:
```
([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], 80)
```

In [381]:
```
len(y_test), critical_region
```

Out[381]:
```
(79, 58)
```

`We got the critical region ∈ [0, 58]. Thus, we could only accept the null hypothesis when we obtain more than 58 correct prediction among total 79 inputs.`

8. Upload to S3

In [382]:
```
# Upload to s3
prefix = 'student_analysis'
data_s3 = sagemaker_session.upload_data(path=os.path.join(sagemaker_dir, exp_dir),
                                        bucket=bucket,
                                        key_prefix=prefix)
data_s3
```

Out[382]:
```
's3://sagemaker-us-east-1-980497197870/student_analysis'
```

9. Create Pytorch estimator & training

```
In [383]: model = PyTorch(entry_point='train.py',
                          source_dir='pytorch_model',
                          role=role,
                          train_instance_count=1,
                          train_instance_type='ml.p2.xlarge',
                          sagemaker_session=sagemaker_session,
                          framework_version='1.5.0',
                          hyperparameters={'epochs': 500,
                                           'input_features': 5,
                                           'hidden_dim': 10,
                                           'output_dim': 2,
                                           'batch-size': 32})

In [384]: model.fit({'train': data_s3})
```

10. Deploy Pytorch estimator

Deploy Pytorch estimator

```
In [385]: predictor = model.deploy(instance_type = 'ml.c4.xlarge',
                                   initial_instance_count = 1)

Parameter image will be renamed to image_uri in SageMaker Python SDK
'create_image_uri' will be deprecated in favor of 'ImageURIProvider'
on SDK v2.

---------------!
```

11. Test model

```
In [26]: def result_pass_fail(preds):
             c = 0
             for idx in range(len(preds)):
                 if (preds[idx] == y_test[idx]).all():
                     c += 1
             return c, 1.0 * c / len(preds)

In [390]: test_data = pd.read_csv(os.path.join(sagemaker_dir, exp_dir, 'test.csv'), header=None, na
          y_test = np.asarray(test_data.iloc[:,[0,1]])
          X_test = test_data.iloc[:,2:]
          X_test = X_test.astype('float32')
          y_preds = np.squeeze(np.rint(predictor.predict(X_test))).astype(int)
          y_preds[:10]

Out[390]: array([[1, 0],
                 [0, 1],
                 [1, 0],
                 [0, 1],
                 [0, 1],
                 [0, 1],
                 [0, 1],
                 [0, 1],
                 [0, 1],
                 [0, 1]])

In [393]: correct, p = result_pass_fail(y_preds)
          print('Correct predictions: ' + str(correct) + ' among the total ' + str(len(y_preds)))
          print('Accuracy score: ' + str(p))

          Correct predictions: 45 among the total 79
          Accuracy score: 0.569620253164557
```

What a surprise! Mom was wrong, those bad activities cannot lead to failing directly.

However, although after predictions, we have to refuse the null hypothesis. I am still shocked by the result. Because we only use 5 background information and we could predict the result with more a half accuracy. So, what if we take all of the background information and try it again?

12.

Experiment 2: Predict result by only use background information. `SetC + Pass/Fail`

Choose data y and data x from local files, then combined them

```
In [27]: # split data into training and test datasets
         X_train, X_test, y_train, y_test = None, None, None, None # make sure X and y are empty
         X_train, X_test, y_train, y_test = train_test_split(setC, outputA, test_size = 0.2, random_st

In [28]: len(X_train), len(y_train), len(X_train.columns), len(y_train.columns)
Out[28]: (316, 316, 26, 2)

In [29]: len(X_test), len(y_test), len(X_test.columns), len(y_test.columns)
Out[29]: (79, 79, 26, 2)
```

13. Upload to S3

```
In [31]: # Upload to s3
         prefix = 'student_analysis'
         data_s3 = sagemaker_session.upload_data(path=os.path.join(sagemaker_dir, exp_dir),
                                                 bucket=bucket,
                                                 key_prefix=prefix)
         data_s3
Out[31]: 's3://sagemaker-us-east-1-980497197870/student_analysis'
```

14. Create Pytorch estimator & training

```
In [27]: model = PyTorch(entry_point='train.py',
                         source_dir='pytorch_model',
                         role=role,
                         train_instance_count=1,
                         train_instance_type='ml.c4.xlarge',
                         sagemaker_session=sagemaker_session,
                         framework_version='1.5.0',
                         hyperparameters={'epochs': 500,
                                          'input_features': 26,
                                          'hidden_dim': 128,
                                          'output_dim': 2,
                                          'batch-size': 32})

In [28]: model.fit({'train': data_s3})
         Epoch: 488, Loss: 0.0006823384141171118
         Epoch: 489, Loss: 0.000398645533277886
         Epoch: 490, Loss: 0.0004143501377257052 6
         Epoch: 491, Loss: 0.0021926929371466 4
```

15. Deploy Pytorch estimator

```
In [29]: predictor = model.deploy(instance_type = 'ml.c4.xlarge',
                                  initial_instance_count = 1)
         Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
         'create_image_uri' will be deprecated in favor of 'ImageURIProvider' class in SageMa
         on SDK v2.

         ---------------!
```

16. Test model

```
In [30]: test_data = pd.read_csv(os.path.join(sagemaker_dir, exp_dir, 'test.csv'), header=None, names=
         y_test = np.asarray(test_data.iloc[:,[0,1]])
         X_test = test_data.iloc[:,2:]
         X_test = X_test.astype('float32')
         y_preds = np.squeeze(np.rint(predictor.predict(X_test))).astype(int)
         y_preds[:10]

Out[30]: array([[1, 0],
                [0, 1],
                [0, 1],
                [0, 1],
                [0, 1],
                [1, 0],
                [0, 1],
                [1, 0],
                [1, 0],
                [0, 1]])

In [31]: correct, p = result_pass_fail(y_preds)
         print('Correct predictions: ' + str(correct) + ' among the total ' + str(len(y_preds)))
         print('Accuracy score: ' + str(p))

         Correct predictions: 58 among the total 79
         Accuracy score: 0.7341772151898734
```

73% accuracy? Only by using background information? Let's find out which features impact the result most. **To do this, train a RandomForest model could help us to find out.**

17. Train a Random Forest model to calculate permutation feature importance

```
In [32]: model = SKLearn(entry_point = 'train.py',
                         source_dir = 'scikit-learn_model',
                         role = role,
                         train_instance_count = 1,
                         train_instance_type = 'ml.c4.xlarge',
                         sagemaker_session = sagemaker_session,
                         framework_version = '0.20.0',
                         hyperparameters = {'n-estimators': 500,
                                            'output-dim': 2})

         This is not the latest supported version. If you would like
         d framework_version=0.23-1 to your constructor.

In [33]: %%time
         model.fit({'train': data_s3})
                    "hosts": [
                        "algo-1"
                    ],
```

18. Test scikit-learn model

```
In [38]: %%time
         predictor = model.deploy(initial_instance_count = 1,
                                  instance_type = 'ml.t2.medium')

         Parameter image will be renamed to image_uri in SageMaker Python SDK v2.

         ----------------!CPU times: user 303 ms, sys: 7.02 ms, total: 310 ms
         Wall time: 8min 32s

In [41]: y_preds = predictor.predict(X_test)

In [44]: accuracy = accuracy_score(y_test, y_preds)
         print(accuracy)

         0.7088607594936709
```

19. Calculate permutation feature importance & plot it

In [51]:
```
model_file_path = 'scikit-learn_model/model.joblib'
```

In [52]:
```
rf1 = joblib.load(model_file_path)
```

In [53]:
```
def r2(rf1, X_train, y_train):
    return r2_score(y_train, rf1.predict(X_train))
```

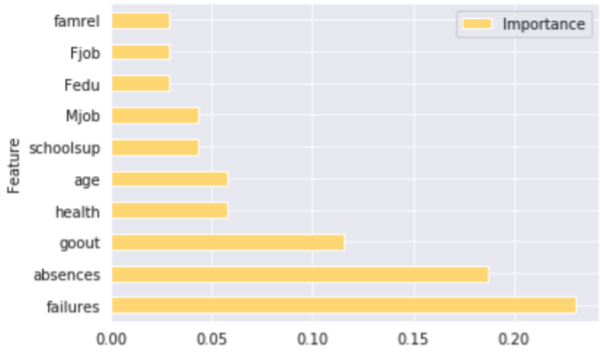
In [56]:
```
prem = permutation_importances(rf1, X_train, y_train, r2)
prem[:10]
```

Out[56]:

Feature	Importance
failures	0.230457
absences	0.187246
goout	0.115229
health	0.057614
age	0.057614
schoolsup	0.043211
Mjob	0.043211
Fedu	0.028807
Fjob	0.028807
famrel	0.028807

In [65]:
```
prem[:10].plot.barh(color='#FDDB7D')
```

Out[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64c92d4940>
```



It seems like 'failures', 'absences' are the most two behaviors that impact students' grades. And 'go_out' which is a bad activity in my mom's eyes would also influence grades a lot. But surprisingly, parents' jobs and 'education of father' also impact a little bit.
'Failures' makes a lot of sense to me, to be honest if you failed lots of tests in the past, you probably will fail other tests in the future. And that opinion indicates we could get better performance if we build our model with past scores(using setB or setA).
'Absences', 'go_out', and 'health' combined together, no matter what reasons cause you to miss the class, maybe you skip the class or caused by personal health status, will impact your grades a lot.
Other behaviors like 'schoolsup'(extra educational support), parents' jobs, parents' education, and 'famrel'(family relationship) point out that environment also very important for students.

As I mentioned above, I believe we could get better results by using past scores. So, in this test, I would use setB as my training dataset.

20. The left four experiments have similarities workflow with these two.

Experiment 3: Can I predict my performance in the next semester at the end of first semester? `SetB + Pass/Fail` ¶

Experiment 4: Can I predict what particular grade will I get in the second semester? `SetB + 5 Level` ¶

Experiment 5: REDO last test by using all informantion. `SetA + 5 Level`

Experiment 6: Could we predict score based on past tests? `SetA + Actual Score` ¶

Performance compared to benchmark models

	Benchmark model (NN)	My model (NN)
Exp1: "bad" + Pass/Fail		0.570
Exp2: SetC + Pass/Fail	0.663 ±0.0010	0.734, 0.709(RF)
Exp3: SetB + Pass/Fail	0.813 ±0.0050	0.873
Exp4: SetB + 5 Level	0.498 ±0.0012	0.595
Exp5: SetA + 5 Level	0.603 ±0.0016	0.721
Exp6: SetA+Actual Score	±2.05 ±0.0020	±1.71

Reference

P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.

This is a demo project for me. Practicing Pytorch, scikit-learn, and AWS sagemaker. Thanks to the original authors! Hope you like this project :)