Project: Midterm Report

Evan Matthews[1], Vikram Ramavarapu[1], and Krishnaveni Unnikrishnan[1]

[1]CS 412 Group G6

November 6th, 2024

Abstract

The internet's pervasive role in modern life has raised concerns about Problematic Internet Use (PIU), particularly among children and teens. Our research aims to predict early signs of PIU using machine learning techniques applied to data from the Child Mind Institute's Healthy Brain Network. The study employs a comprehensive methodology combining cross-sectional and time-series data analysis. Initial results from multiple models, including Random Forest, XGBoost, SVM, and Feed Forward Neural Networks, demonstrate promising accuracy rates, with XGBoost achieving the highest mean accuracy of 0.6821. Our project experimentation is structured in three phases: data preprocessing, initial model evaluation, and fine-feature reevaluation. The methodology incorporates innovative approaches such as sequential modeling for time-series data and ensemble techniques combining cross-sectional and sequential models. Preliminary findings suggest that machine learning can effectively predict PIU severity using quantitative measures rather than traditional subjective assessments. This research contributes to the growing field of digital health by providing a data-driven approach to identifying at-risk youth for early intervention.

1 Introduction, Motivation

The internet has become an integral part of our daily lives, with people of all ages spending a significant amount of time online. This trend has given rise to concerns about the potential impacts of excessive internet use, particularly on children and teens. Problematic Internet Use (PIU) is a condition characterized by excessive or poorly controlled preoccupations, urges, or behaviors regarding computer use and internet access that lead to impairment or distress [4]. PIU has been associated with a range of mental health issues, including depression, anxiety, and impulsivity [2]. As such, identifying early signs of PIU in children and teens is crucial for prevention and intervention. In this project, we aim to predict early signs of PIU in children and teens using machine learning techniques, leveraging data from the Child Mind Institute's Healthy Brain Network. The project plan consists of three phases: data preprocessing, initial model evaluation, and fine-feature reevaluation. We will submit our work to the Child Mind Institute's (CMI) Kaggle competition on PIU prediction, and we also aim to publish our results as a paper should they outperform competition expectations.

Despite having multiple studies showing the negative effects of excessive internet use, exact details about PIU warning signs and the most at-risk individuals are still unknown. These studies can be useful, but their results focus primarily on written or binary feedback from students or parents. These studies can be useful, but they also introduce biases and often fail to show the true factors which correlate a participant's estimated internet impact [1, 5].

2 Related work, Scope

Research on Problematic Internet Use (PIU) has gained significant attention due to its increasing prevalence and association with various psychological and behavioral issues. Early investigations into PIU highlighted its similarities with substance use disorders, impulse control disorders, and obsessive-compulsive disorder.

Studies have revealed concerning prevalence rates between 1.5% and 8.2% in the United States and Europe, emphasizing the growing social impact of this condition. The relationship between PIU and psychiatric disorders has been extensively documented, with research showing significant associations with depressive disorders and attention-deficit/hyperactivity disorder (ADHD). A notable study found that individuals with PIU were more than twice as likely to have depressive disorders ($aOR = 2.43$), and showed increased likelihood of having ADHD combined presentation ($aOR = 1.91$) and Autism Spectrum Disorder ($aOR = 2.24$).

Recent investigations have focused on understanding the personality profiles and emotional factors contributing to PIU. Research has identified specific personality traits associated with PIU, including lower scores in novelty seeking, harm avoidance, and reward dependence. Additionally, emotional dysregulation has emerged as a significant factor, with studies suggesting that PIU may serve as a behavioral mechanism for escaping negative affects.

Treatment approaches for PIU have primarily centered on addressing comorbid conditions, with cognitive behavioral therapy and selective serotonin reuptake inhibitors showing promise as potential interventions. However, researchers emphasize that detailed treatment guidelines require further investigation, particularly given the complex interplay between PIU and various psychological disorders.

The field continues to evolve, with ongoing debates about diagnostic criteria and classification. While the Internet's positive impact on well-being is widely acknowledged, the pathological aspects of its use remain understudied, particularly regarding subtle psychological changes such as online disinhibition. This highlights the need for additional research into the pathophysiology, epidemiology, natural course, and treatment of PIU to develop more effective intervention strategies.

In terms of our current work, given that the original scope of the project was accepted, we are pressing forward with this plan with no significant changes. The most crucial critique provided- that the validation plan and evaluation metric were not clear- are likewise addressed in the methodology section.

3 Methodology

Data for this project has two components: cross-sectional, and time-series. The cross-sectional data is per participant and contains fields described in the following table. Each time-series dataset is per participant and each entry of the dataset represents the status of the participant's heartrate monitor at a given point in time. PCIAT is the Parent-Child Internet Addiction Test score, which is used to compute the Severity of Internet Addiction Index (SII) score. The SII score is the target variable for this project. For the description of fields in the time-series dataset, see Table 6.

The project is divided into three phases: data preprocessing, initial model evaluation, and fine-feature reevaluation. The data preprocessing phase entails dropping survey-based fields used to compute PCIAT, which is then used to compute the SII, as our model's intention is to compute SII directly from the other metrics. Missing values in the data are filled using iterative imputation, and the missing SII values are filled in using K-Nearest Neighbors (k=5).

Multiple models will be evaluated on the cross-sectional data: Random Forest, XGBoost, SVM, and a feed forward neural network. After this, a sequential model, evaluated amongst transformers or auto-encoders, will be trained on the time-series data. The sequential model will allow us to compute an embedding of the time-series data, which will be used as an additional feature in the cross-sectional model. The final model will be an ensemble of the cross-sectional and sequential models, with the sequential model's embedding as an additional feature in the cross-sectional model. The classifier model will be retrained on the concatenated dataset, to predict the SII.

4 Preliminary Results

The results of the 10-fold cross-validation provide insight into the performance consistency of each model - Random Forest Classifier, XGBoost Classifier, Support Vector Classifier, and Feed Forward Neural Network across different subsets of the dataset. This method helps ensure that the reported accuracy is not overly dependent on any particular training subset, giving a more reliable view of how each model would perform in a real-world setting. These results are summarized in Table 1 and **??**.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
RF	0.684	0.682	0.682	0.674	0.687	0.674	0.684	0.707	0.649	0.672
XGB	0.689	0.667	0.669	0.689	0.682	0.684	0.674	0.732	0.636	0.694
SVC	0.649	0.657	0.646	0.649	0.649	0.646	0.649	0.649	0.654	0.652
FFN	0.710	0.649	0.669	0.692	0.689	0.684	0.687	0.674	0.649	0.694

Table 1: 10-Fold Cross-Validation Results for Each Model

Using hypothesis testing we conclude that XGBoost model is significantly better than the other models based on a Student's t-test at significance level $\alpha = 5\%$ and number of parameters to be trained.

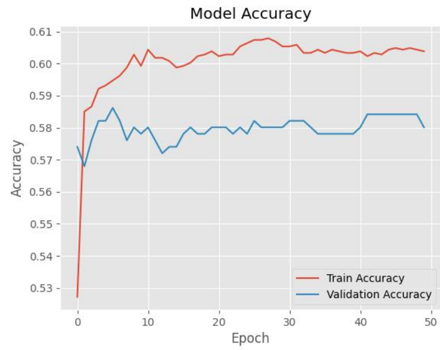

Figure 1: Model accuracy of Feed-Foward Neural Network

Model	**Mean Accuracy**
Random Forest Classifier	0.680
XGBoost Classifier	0.682
Support Vector Classifier	0.650
Feed Forward Neural Network	0.680

Table 2: Mean Accuracy for each model

5 Plan of Work

The next steps entail training and evaluating different sequential models, amongst transformers and auto-encoders on the time-series data. The best model will be selected based on performance metrics, and the model will be used to compute an embedding of the time-series data. The embedding will be used as an additional set of features in the cross-sectional model. The final model will be an ensemble of the cross-sectional and sequential models, with the sequential model's embedding as an additional feature in the cross-sectional model. The classifier model will be retrained on the concatenated dataset, to predict the SII.

Alternatively, an approach where PCIAT scores are preliminarily computed and then used to compute the SII score can be evaluated. This approach would be evaluated against the sequential model approach, and the best approach will be selected based on performance metrics.

6 Conclusions, discussions

With our current progress and plan forward in mind, we are confident in our ability to complete our project on time and with a high degree of success. Our initial results show promise given our experiments with both a feed-forward neural and XGB classification. We are also excited to see how our subsequent models will perform with respect to our current baseline accuracy. Predictions with respect to time-series data are expected to greatly outperform our current models, but only time will validate this hypothesis.

References

[1] Elias Aboujaoude. Problematic internet use: an overview. *World Psychiatry*, 9(2):85–90, June 2010.

[2] Hilarie Cash, Cosette D Rae, Ann H Steel, and Alexander Winkler. Internet addiction: A brief summary of research and practice. *Curr. Psychiatry Rev.*, 8(4):292–298, November 2012.

[3] Antonina Dolgorukova. Cmi-piu: Features eda, Nov 2024.

[4] Mauro Pettorruso, Stephanie Valle, Elizabeth Cavic, Giovanni Martinotti, Massimo di Giannantonio, and Jon E Grant. Problematic internet use (PIU), personality profiles and emotion dysregulation in a cohort of young adults: trajectories from risky behaviors to addiction. *Psychiatry Res.*, 289(113036):113036, July 2020.

[5] Anita Restrepo, Tohar Scheininger, Jon Clucas, Lindsay Alexander, Giovanni A Salum, Kathy Georgiades, Diana Paksarian, Kathleen R Merikangas, and Michael P Milham. Problematic internet use in children and adolescents: associations with psychiatric disorders and impairment. *BMC Psychiatry*, 20(1):252, May 2020.

Appendix A: Feature Importance, Feed-Forward Neural Network

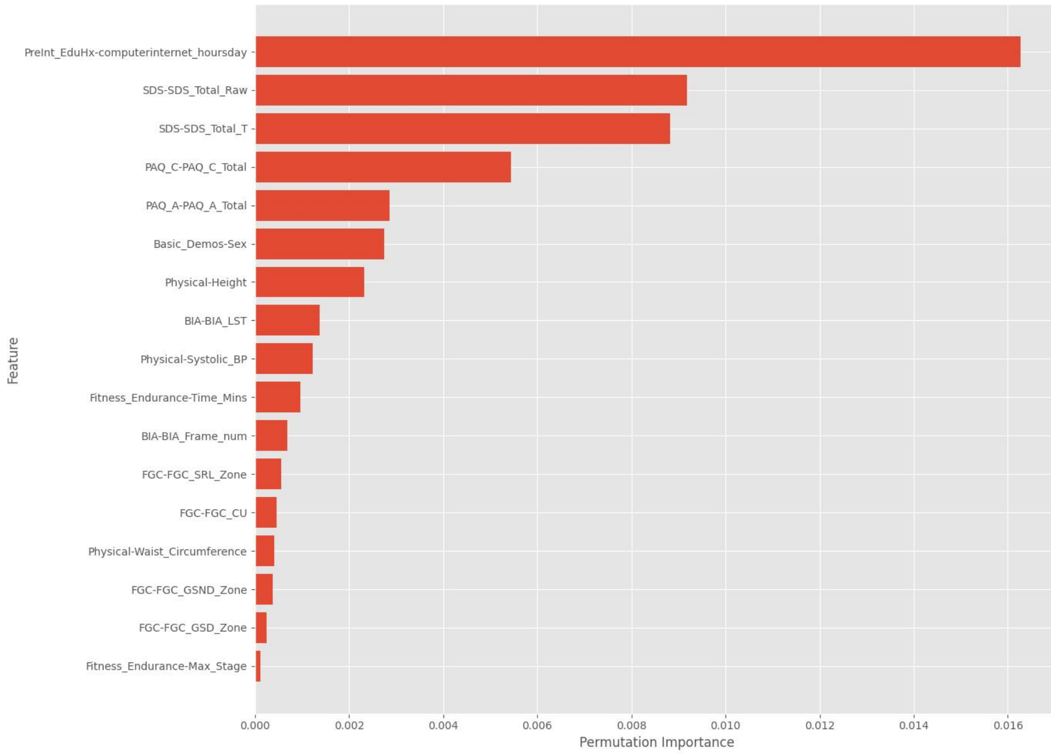

Figure 2: Features plotted in order of importance, ascending.

Appendix B: Description of Fields in Time-Series Dataset

Field	**Description**
step	Step count
X	X-axis acceleration of the heartrate monitor
Y	Y-axis acceleration of the heartrate monitor
Z	Z-axis acceleration of the heartrate monitor
enmo	Euclidean Norm Minus One (ENMO)
anglez	Angle in the Z-axis
non-wear_flag	Non-wear flag
light	Light exposure
battery_voltage	Battery voltage of the monitor
time_of_day	Time of day
weekday	Day of the week
quarter	Quarter of the year
relative_date_PCIAT	Current PCIAT minus previous day PCIAT

Appendix C: Kaggle Starter Code [3]

```
# Starter Notebook: Multi-Target Prediction Using CatBoost

See [this discussion](https://www.kaggle.com/competitions/child-mind
    -institute-problematic-internet-use/discussion/535121) for more
    information.
import warnings
from functools import partial
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np
import optuna
import polars as pl
import polars.selectors as cs
from catboost import CatBoostRegressor, MultiTargetCustomMetric
from numpy.typing import ArrayLike, NDArray
from polars.testing import assert_frame_equal
from sklearn.base import BaseEstimator
from sklearn.metrics import cohen_kappa_score
from sklearn.model_selection import StratifiedKFold

warnings.filterwarnings("ignore", message="Failed to optimize method
    ")

DATA_DIR = Path("./child-mind-institute-problematic-internet-use")
TARGET_COLS = [
    "PCIAT-PCIAT_01",
    "PCIAT-PCIAT_02",
    "PCIAT-PCIAT_03",
    "PCIAT-PCIAT_04",
    "PCIAT-PCIAT_05",
    "PCIAT-PCIAT_06",
    "PCIAT-PCIAT_07",
    "PCIAT-PCIAT_08",
    "PCIAT-PCIAT_09",
    "PCIAT-PCIAT_10",
    "PCIAT-PCIAT_11",
    "PCIAT-PCIAT_12",
    "PCIAT-PCIAT_13",
    "PCIAT-PCIAT_14",
    "PCIAT-PCIAT_15",
    "PCIAT-PCIAT_16",
    "PCIAT-PCIAT_17",
    "PCIAT-PCIAT_18",
```

```
    "PCIAT-PCIAT_19",
    "PCIAT-PCIAT_20",
    "PCIAT-PCIAT_Total",
    "sii",
]

FEATURE_COLS = [
    "Basic_Demos-Enroll_Season",
    "Basic_Demos-Age",
    "Basic_Demos-Sex",
    "CGAS-Season",
    "CGAS-CGAS_Score",
    "Physical-Season",
    "Physical-BMI",
    "Physical-Height",
    "Physical-Weight",
    "Physical-Waist_Circumference",
    "Physical-Diastolic_BP",
    "Physical-HeartRate",
    "Physical-Systolic_BP",
    "Fitness_Endurance-Season",
    "Fitness_Endurance-Max_Stage",
    "Fitness_Endurance-Time_Mins",
    "Fitness_Endurance-Time_Sec",
    "FGC-Season",
    "FGC-FGC_CU",
    "FGC-FGC_CU_Zone",
    "FGC-FGC_GSND",
    "FGC-FGC_GSND_Zone",
    "FGC-FGC_GSD",
    "FGC-FGC_GSD_Zone",
    "FGC-FGC_PU",
    "FGC-FGC_PU_Zone",
    "FGC-FGC_SRL",
    "FGC-FGC_SRL_Zone",
    "FGC-FGC_SRR",
    "FGC-FGC_SRR_Zone",
    "FGC-FGC_TL",
    "FGC-FGC_TL_Zone",
    "BIA-Season",
    "BIA-BIA_Activity_Level_num",
    "BIA-BIA_BMC",
    "BIA-BIA_BMI",
    "BIA-BIA_BMR",
    "BIA-BIA_DEE",
    "BIA-BIA_ECW",
    "BIA-BIA_FFM",
```

```
    "BIA-BIA_FFMI",
    "BIA-BIA_FMI",
    "BIA-BIA_Fat",
    "BIA-BIA_Frame_num",
    "BIA-BIA_ICW",
    "BIA-BIA_LDM",
    "BIA-BIA_LST",
    "BIA-BIA_SMM",
    "BIA-BIA_TBW",
    "PAQ_A-Season",
    "PAQ_A-PAQ_A_Total",
    "PAQ_C-Season",
    "PAQ_C-PAQ_C_Total",
    "SDS-Season",
    "SDS-SDS_Total_Raw",
    "SDS-SDS_Total_T",
    "PreInt_EduHx-Season",
    "PreInt_EduHx-computerinternet_hoursday",
]
# Load data
train = pl.read_csv(DATA_DIR / "train.csv")
test = pl.read_csv(DATA_DIR / "test.csv")
train_test = pl.concat([train, test], how="diagonal")

IS_TEST = test.height <= 100

assert_frame_equal(train, train_test[: train.height].select(train.
   columns))
assert_frame_equal(test, train_test[train.height :].select(test.
   columns))
# Cast string columns to categorical
train_test = train_test.with_columns(cs.string().cast(pl.Categorical
   ).fill_null("NAN"))
train = train_test[: train.height]
test = train_test[train.height :]

# ignore rows with null values in TARGET_COLS
train_without_null = train_test.drop_nulls(subset=TARGET_COLS)
X = train_without_null.select(FEATURE_COLS)
X_test = test.select(FEATURE_COLS)
y = train_without_null.select(TARGET_COLS)
y_sii = y.get_column("sii").to_numpy()  # ground truth
cat_features = X.select(cs.categorical()).columns
class MultiTargetQWK(MultiTargetCustomMetric):
    def get_final_error(self, error, weight):
        return np.sum(error)  # / np.sum(weight)
```

```
    def is_max_optimal(self):
        # if True, the bigger the better
        return True

    def evaluate(self, approxes, targets, weight):
        approx = np.clip(approxes[-1], 0, 3).round().astype(int)
        target = targets[-1]

        qwk = cohen_kappa_score(target, approx, weights="quadratic")

        return qwk, 1

    def get_custom_metric_name(self):
        return "MultiTargetQWK"


class OptimizedRounder:
    """
    A class for optimizing the rounding of continuous predictions
       into discrete class labels using Optuna.
    The optimization process maximizes the Quadratic Weighted Kappa
       score by learning thresholds that separate
    continuous predictions into class intervals.

    Args:
        n_classes (int): The number of discrete class labels.
        n_trials (int, optional): The number of trials for the
           Optuna optimization. Defaults to 100.

    Attributes:
        n_classes (int): The number of discrete class labels.
        labels (NDArray[np.int_]): An array of class labels from 0
           to `n_classes - 1`.
        n_trials (int): The number of optimization trials.
        metric (Callable): The Quadratic Weighted Kappa score metric
            used for optimization.
        thresholds (List[float]): The optimized thresholds learned
           after calling `fit()`.

    Methods:
        fit(y_pred: NDArray[np.float_], y_true: NDArray[np.int_]) ->
            None:
            Fits the rounding thresholds based on continuous
               predictions and ground truth labels.

            Args:
                y_pred (NDArray[np.float_]): Continuous predictions
```

```
                that need to be rounded.
            y_true (NDArray[np.int_]): Ground truth class labels
                .

        Returns:
            None

    predict(y_pred: NDArray[np.float_]) -> NDArray[np.int_]:
        Predicts discrete class labels by rounding continuous
            predictions using the fitted thresholds.
        `fit()` must be called before `predict()`.

        Args:
            y_pred (NDArray[np.float_]): Continuous predictions
                to be rounded.

        Returns:
            NDArray[np.int_]: Predicted class labels.

    _normalize(y: NDArray[np.float_]) -> NDArray[np.float_]:
        Normalizes the continuous values to the range [0, `
            n_classes - 1`].

        Args:
            y (NDArray[np.float_]): Continuous values to be
                normalized.

        Returns:
            NDArray[np.float_]: Normalized values.

References:
    - This implementation uses Optuna for threshold optimization
        .
    - Quadratic Weighted Kappa is used as the evaluation metric.
"""

def __init__(self, n_classes: int, n_trials: int = 100):
    self.n_classes = n_classes
    self.labels = np.arange(n_classes)
    self.n_trials = n_trials
    self.metric = partial(cohen_kappa_score, weights="quadratic
        ")

def fit(self, y_pred: NDArray[np.float_], y_true: NDArray[np.
    int_]) -> None:
    y_pred = self._normalize(y_pred)
```

```
        def objective(trial: optuna.Trial) -> float:
            thresholds = []
            for i in range(self.n_classes - 1):
                low = max(thresholds) if i > 0 else min(self.labels)
                high = max(self.labels)
                th = trial.suggest_float(f"threshold_{i}", low, high
                    )
                thresholds.append(th)
            try:
                y_pred_rounded = np.digitize(y_pred, thresholds)
            except ValueError:
                return -100
            return self.metric(y_true, y_pred_rounded)

        optuna.logging.disable_default_handler()
        study = optuna.create_study(direction="maximize")
        study.optimize(
            objective,
            n_trials=self.n_trials,
        )
        self.thresholds = [study.best_params[f"threshold_{i}"] for i
            in range(self.n_classes - 1)]

    def predict(self, y_pred: NDArray[np.float_]) -> NDArray[np.int_
        ]:
        assert hasattr(self, "thresholds"), "fit() must be called
            before predict()"
        y_pred = self._normalize(y_pred)
        return np.digitize(y_pred, self.thresholds)

    def _normalize(self, y: NDArray[np.float_]) -> NDArray[np.float_
        ]:
        # normalize y_pred to [0, n_classes - 1]
        return (y - y.min()) / (y.max() - y.min()) * (self.n_classes
            - 1)
# setting catboost parameters
params = dict(
    loss_function="MultiRMSE",
    eval_metric=MultiTargetQWK(),
    iterations=1 if IS_TEST else 100000,
    learning_rate=0.1,
    depth=5,
    early_stopping_rounds=50,
)

# Cross-validation
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=52)
```

```
models: list[CatBoostRegressor] = []
y_pred = np.full((X.height, len(TARGET_COLS)), fill_value=np.nan)
for train_idx, val_idx in skf.split(X, y_sii):
    X_train: pl.DataFrame
    X_val: pl.DataFrame
    y_train: pl.DataFrame
    y_val: pl.DataFrame
    X_train, X_val = X[train_idx], X[val_idx]
    y_train, y_val = y[train_idx], y[val_idx]

    # train model
    model = CatBoostRegressor(**params)
    model.fit(
        X_train.to_pandas(),
        y_train.to_pandas(),
        eval_set=(X_val.to_pandas(), y_val.to_pandas()),
        cat_features=cat_features,
        verbose=False,
    )
    models.append(model)

    # predict
    y_pred[val_idx] = model.predict(X_val.to_pandas())

assert np.isnan(y_pred).sum() == 0
# Optimize thresholds
optimizer = OptimizedRounder(n_classes=4, n_trials=300)
y_pred_total = y_pred[:, TARGET_COLS.index("PCIAT-PCIAT_Total")]
optimizer.fit(y_pred_total, y_sii)
y_pred_rounded = optimizer.predict(y_pred_total)

# Calculate QWK
qwk = cohen_kappa_score(y_sii, y_pred_rounded, weights="quadratic")
print(f"Cross-Validated QWK Score: {qwk}")
feature_importance = np.mean([model.get_feature_importance() for
   model in models], axis=0)
sorted_idx = np.argsort(feature_importance)
fig = plt.figure(figsize=(12, 10))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx],
   align="center")
plt.yticks(range(len(sorted_idx)), np.array(X_test.columns)[
   sorted_idx])
plt.title("Feature Importance")
class AvgModel:
    def __init__(self, models: list[BaseEstimator]):
        self.models = models
```

```
    def predict(self, X: ArrayLike) -> NDArray[np.int_]:
        preds: list[NDArray[np.int_]] = []
        for model in self.models:
            pred = model.predict(X)
            preds.append(pred)

        return np.mean(preds, axis=0)
avg_model = AvgModel(models)
test_pred = avg_model.predict(X_test.to_pandas())[:, TARGET_COLS.
   index("PCIAT-PCIAT_Total")]
test_pred_rounded = optimizer.predict(test_pred)
test.select("id").with_columns(
    pl.Series("sii", pl.Series("sii", test_pred_rounded)),
).write_csv("submission.csv")
```

Appendix D: Hypothesis Testing

In order to compare two models A and B the null hypothesis is that the distribution of $acc(A)_i - acc(B)_i$ has zero mean and the alternative hypothesis is that the model with the higher mean performance accuarcy is significantly better than the other.
T statistic is given by:

$$t = \frac{\overline{acc}(A) - \overline{acc}(B)}{\sqrt{var(A-B)/k}}$$

where,

$$var(A-B) = \frac{1}{k}\sum_{i=1}^{k}[acc(A)_i - acc(B)_i - (\overline{acc}(A) - \overline{acc}(B))]^2$$

Model 1	**Model 2**	**t-statistic**	**p-value**	**Accept/Reject Null**
RF	XGB	-0.492	0.633	Accept
RF	SVC	6.149	0.0	Reject
RF	FFN	-0.039	0.970	Accept
XGB	FFN	0.304	0.767	Accept
XGB	SVC	4.104	0.002	Reject
FFN	SVC	4.588	0.001	Reject

Table 3: Comparison of Models using t-statistic and p-value

Therefore, we can conclude that RF, XGB and FFN are significantly better than SVC. However it looks like the distribution of accuracies of RF , XGB and FFN are comparable. In this case we will choose the model that requires the least amount of parameters which is the XGBoost classifier.